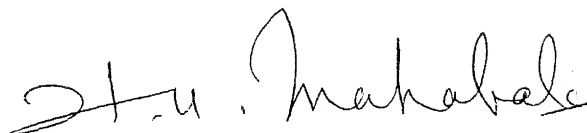


CERTIFICATE

Certified that this work on "the Design of  
an Educational Digital Computer" has been carried out under my  
supervision and that this has not been submitted elsewhere for  
a degree,

A handwritten signature in cursive script, reading "H.N. Mahabala".

Dr. H.N. Mahabala  
Assistant Professor  
Department of Electrical Engineering  
Indian Institute of Technology, Kanpur

August ,1968

### ACKNOWLEDGEMENTS

The author is deeply indebted to Dr. H.N.Mahabala for his able guidance and encouragement throughout the course of this project. He also takes this opportunity to thank Dr. T.R.Viswanathan, Dr. V.Rajaramn and Professor D.L.Stephenson for making valuable suggestion.

Finally he owes thanks to CSIR for providing financial assistance for this project.

## SYNOPSIS

Design of an Educational Digital Computer

Chandra Vir Singh

✓ This report describes the system and logical design of an Educational Digital Computer EDC. EDC is a small, low cost computer to be made from indigenous components, and is meant for educational purposes in a digital laboratory. The main features of the machine are a 12 bit 256 words magnetic core memory, parallel operation, synchronus control and I/O through a teleprinter.

A general logic circuit simulation program written in Fortran IV language to check the design of any digital system is also presented,

## Table of Contents

Chapter		Page No.
1	INTRODUCTION	1
2	System Design	3
2.1	Machine features of EDC	3
2.2	Number of & types of instructions	6
2.3	Word format	8
2.4	System Organization	9
2.5	Functional Description	9
3.	Logical Design	15
3.1	Logical Symbolism and diagrams	15
3.2	Instruction Codes and their control wires	15
3.3	Arithmetic Unit	20
3.4	Control Unit	24
3.5	Input Output Control Unit	31
3.6	The Console	32
4.	Choice of Logic family	35
4.1	Different Logic Circuit configurations	35
4.2	RTL, DTL, and TTL Logic Circuits	35
5.	Digital Logic Simulation	41
5.1	Need of Logical Check	41
5.2	Program Characteristics	42
5.3	Simulation Technique	42
5.4	Computer Program Structure	46

61	Conclusions	47
	References	48
	<u>Appendix</u>	
1	Reference Manual	49
2	System Flow Chart	51
3	Program Listing	56
4	Sample Program	79
5	List of Symbols	97

## CHAPTER 1

### INTRODUCTION

There has been a rapid growth in the number of the computers in the industrial and research organization of the country. This requires that educational facilities in computer technology be available in the educational institutions. A well equipped digital laboratory would be an essential and integral part of such a program.

A digital laboratory needs to have a set of logical modules, which form a universal set and can be used to fabricate any digital system. A small low cost computer, made from indigenous components, and having a modular structure so that it may be easily dismantled and reassembled, would be highly desirable in such a laboratory.

The design of modules that will form the building blocks for the computer, should take into account factors such as cost, environmental conditions and availability of the components. This requires that a survey be made of the components available in the country and then determining which logic family is most suited to the needs.

The system design, which determines the basic organization of the computer should take into account factors such as simplicity of basic logical design, ease of programming, efficient utilization of logical elements and simplicity of maintenance. An elegant and simple structure for the computer is an essential requirement, meant for educational purposes. Once the building block has been made a logical design is carried out to produce a set of wiring diagrams which show how to connect the building blocks to realize the specifications set by the system design.

Making a computer is a costly proposition. It is therefore, necessary to carry out exhaustive testing of logical design before producing a prototype for final fabrication. The testing of logical design can best be carried out by simulating the logical circuits on a computer. A general logic circuit simulation program to which description of any logic circuit could be input as data would be most desirable. This simulator can be used for testing the various functional units separately and then the system as a whole.

The purpose of this report is to describe the system design and logical design of a small educational computer. A general logic circuit simulation program is also given.

System design of proposed Educational Digital Computer (EDC) is presented in this chapter. Design considerations and choice of machine features are discussed in detail. This chapter also includes a detailed description of system organization, functional units, word format, instruction list and system specifications.

## 2.1 Machine features of EDC:

Basic features of machine have been chosen after considering the following factors<sup>1</sup>.

### 2.1.1 Serial or Parallel operation

#### (a) Circuitry:

- (i) Cost:- Addition in a parallel machine is performed simultaneously for all the bits. Hence separate adder circuit is provided for each bit. Serial machine on the other hand uses one adder and addition is done bit by bit. The difference in cost of circuit between parallel and serial machines is greater for machines with a large word length. Hence parallel machines are more expensive.
- (ii) Complexity of Control:- The timing sequence is simple in parallel machine because information is transferred within different units of computer by application of a single pulse to several gates. In a serial machine a set of timing pulses corresponding to individual bit positions must be generated and applied to various gates.

(b) Speed:

Parallel machines are faster than serial ones. Basic unit of time in parallel machine is chosen on the basis of addition of two numbers. In serial machine transfer of word from one unit to another determines the basic clock time.

(c) Compatibility with type of Control:

Serial machine is compatible with synchronus control only whereas parallel machine can have either synchronus or asynchronus control.

2.1.2 Synchronus or Asynchronus Operation:

A method of control in which the sequence of commands is generated by means of a master clock followed by timing circuits, is referred to as synchronus control. The timing of all operations are controlled or synchronized with the clock and each operation requires an integral number of clock intervals. A method of control in which a start signal causes certain action to be taken and its completion generates signal to initiate the proper following action is referred to as asynchronus control. In an asynchronus computer there is no fixed time reference, each operation being initiated as soon as the previous one is completed.

(a) Speed of operation:

In a synchronus system slowest operation determines the operational period. Asynchronus operation gives an advantage in speed because the average time to transmit a signal through a network is average delay per circuit multiplied by the number of circuits.

(b) Component Tolerance:

In synchronus system only limited tolerance is allowed because of requirements on waveshapes and pulse coincidence at gates. Asynchronus system are operable with components of considerable tolerances.

Most of the computers are asynchronous in nature and this may be ascribed to several factors including the following.

- (1) There are potential hazards in asynchronous network due to variation in response of active elements and in transmission time of signals.
- (2) Hazard free operation in asynchronous systems are considered more difficult to design, and service.

### 2.1.3 Numerical representation in the Arithmetic Unit:

#### (a) Number base:

Arithmetic operations are more complex in BCD representation.

Binary arithmetic operations are much simpler.

#### (b) Radix Point:

Arithmetic unit is simpler in the case of fixed point arithmetic but has the disadvantage of limited range of numbers and manual tracking of Q point. Floating point arithmetic has got wide range of numbers but the arithmetic unit is complex.

#### (c) Representation of negative numbers:

- (i) Sign and Magnitude:- Multiplication and division is easy and input output data conversion is simpler. Addition and subtraction is more complex than with complement representation. There are two values of zero. Separate sign circuit is provided to determine sign of different operations.
- (ii) Two's Complement:- Exactly one cycle is required for addition and subtraction of any 2's complement number. Obtaining a complement is easy. Extra hardware is required for multiplication and division. Overflow detection is simpler and

representation of zero is unique.

- (iii) One's Complement:- Addition and subtraction requires an extra cycle for end around carry. This slows down the speed. Representation of zero is not unique and overflow detection is difficult in comparison to 2's complement representation.

On the basis of above discussion on different design factors, following machine features are chosen for proposed system EDC

- i binary fixed point machine
- ii parallel operation
- iii synchronus control
- iv two's complement arithmetic
- v single address machine

## 2.2 Number and type of instructions:

The following instructions are chosen for EDC

<u>Octal Code</u>	<u>Mnemonic Symbol</u>	<u>Operation</u>
-------------------	------------------------	------------------

### (a) Arithmetic Instructions

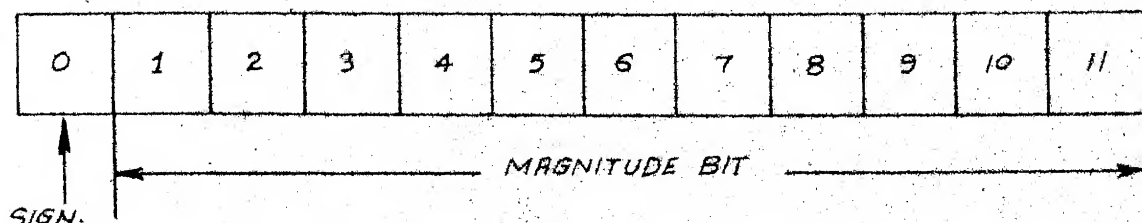
- |    |   |       |                                       |
|----|---|-------|---------------------------------------|
| 1. | 1 | CIA m | Clear the accumulator and add m to it |
| 2. | 2 | ADD m | Add m to the accumulator              |
| 3. | 3 | SUB m | Subtract m from the accumulator       |

### (b) Storage Instructions

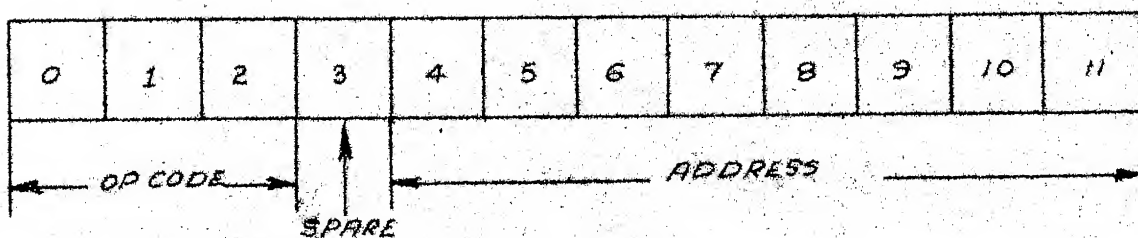
- |    |   |       |  |
|----|---|-------|--|
| 4. | 4 | STO m | Contents of accumulator is stored in m |
|----|---|-------|--|

### (c) Transfer and Control Instructions

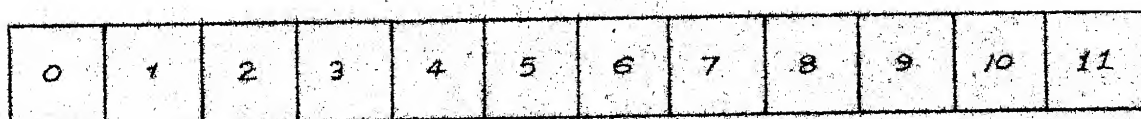
- |    |   |       |   |
|----|---|-------|---|
| 5. | 5 | TRA m | Transfer control to storage location m                                    |
| 6. | 6 | TRM m | Transfer control to storage location m if sign of Accumulator is negative |
| 7. | 7 | HLT   | Stop  |



(a) NUMBER



(b) INSTRUCTION



BIT 0,1,2 OP CODE

3,7-11 SPARE

4 PRESENCE OR ABSENCE OF I/O INSTRUCTION

5,6 OG OUT PUT OCTAL / IO INPUT OCTAL

01 " BED / 11 " BED

(c) I/O MICRO-INSTRUCTION

FIG. 2-1. WORD FORMAT.



## 2.4 System Organization:

A system block diagram showing major functional blocks within each of the functional units is drawn in Figure 2.2. Memory buffer register (MBR ) and memory address register (MAR) are included in the memory unit. All arithmetic operations are performed in the accumulator using Accumulator as an operand. MBR is used as the second register needed in arithmetic operations. Instruction counter (IC) holds the address of next instruction to be executed. Instruction register (IR) holds the instruction in execution cycle. Input-output to the computer is through the teleprinter. I/O buffer register holds the information before it is transmitted to and from the accumulator. Input and output is in either octal or BCD character. Teleprinter I/O is under programme control.

## 2.5 Functional Description:

### 2.5.1 Memory Unit<sup>2</sup>

The memory storage consists of 13 bit, 256 word (16x16) ccm stack. Information transmission between memory unit and other unit of the computer is through MBR in parallel mode.

Memory Buffer Register(MBR):- It is a 13 bit register. Only 12 bits of MBR is used with CU and AU . One bit is kept as a spare. Information is read from a memory cell into MBR and rewritten into the cell in one memory cycle time. The 13th bit is used for parity inside the memory.

Memory Address Register (MAR):- It is a 8 bit register containing address of the memory location currently selected for reading or writing. All 256 words of core memory are addressable by this register.

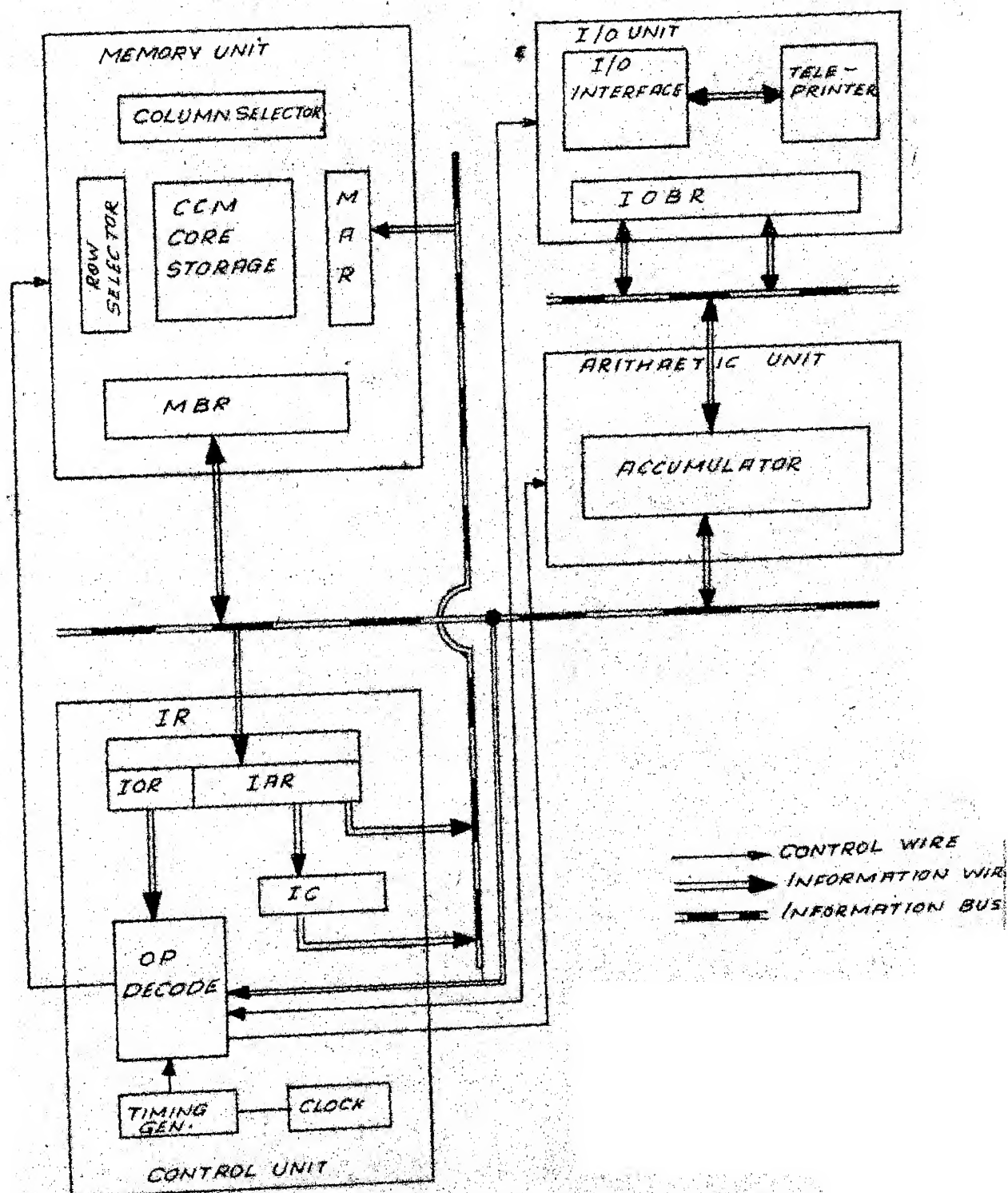


FIG. 2-2. SYSTEM EDC BLOCK DIAGRAM

Parity Checking:- A parity bit detector checks the information as it is removed from the memory. A parity bit generator generates an extra bit according to the rules of the parity just before a word is written into the memory. 13th bit of memory cell is used for parity check.

Characteristics of MU:-

Storage capacity	256 words of 12 bit length
Access Time	3 $\mu$ sec.
Memory cycle time	10 $\mu$ sec.

Input lines to the MU:-

- 8 address lines of MAR
- 2 lines to READ/WRITE circuit
- 1 initiate line to the timing generator
- 12 word bit lines to the MBR

Output lines:-

- 12 word bit lines from the MBR
- 1 parity check line
- 1 line for the OP complete signal from the timing generator

2.5.2 Input-Output Unit<sup>3</sup>:

An overall functional block diagram of the I/O unit and its interaction with the rest of the computer is shown in Fig. 2.2. Reading-in and printing-out information are under programme control. I/O channel is connected to AC.

- Information lines from main computer unit to the I/O unit
- 2 lines READ/WRITE information
- 2 lines for mode selection (octal or BCD)
- 12 word bit lines to IOBR

Information lines from I/O unit to the main computer.

OP complete line-OPC

12 word bit lines from IOBR to AC

### 2.5.3 Arithmetic Unit:

Twelve bit adder is the basic unit of the AU. It works in conjunction with MBR and AC. Result of addition is stored back in the AC. Subtraction is performed by the addition of complement of contents of MBR to AC. All transfer operation between AC and MBR; AC and IOBR is in parallel mode.

Accumulator:- It is a 12 bit register. All arithmetic operations are performed in this register. It can be cleared, content of MBR can be added to its contents and its content can be transferred to MBR and IOBR.

### 2.5.4 Control Unit:

The control unit consists of an instruction register which accepts instruction words from memory, and control circuits which sequence, translate and execute these instructions by means of a set of sub commands.

Sequencing is accomplished jointly by the memory and control units in a basic machine cycle. The control unit is provided with a counter (IC) which is always set to the address of next instruction to be executed.

Translation is performed by the decoding circuit. It generates the actual electrical signals for execution of each of the instructions from the IR. It develops a sequence of detailed commands to transfer the information withing the arithmetic unit and other units of the computer.

Execution depends on timing pulses as well as subcommands. The control must produce as many sequences of gating or switching signals as

the number of different arithmetic, logical and transfer operations the computer is required to perform. It must also be capable of assuming at least as many logical configuration (i.e., binary states) as there are different steps required for the execution of all these instructions. This sets the lower limit to the number of active elements in the control unit.

Manual control is available to allow the operator to intervene, monitor, or modify the automatic operation of the computer.

Instruction Counter:- It is a 8 bit register and contains the address of the memory which contains the next instruction to be executed. Information enters into IC from IAR or switch register on console. Contents of IC are transferred to MAR for address selection. An adder is attached to IC to increment its contents by one.

#### Instruction Mode:-

Fetch State:- Upon completion of an instruction, the instruction mode flipflop is set to FETCH state. It allows the address stored in the IC to choose the next instruction word from memory to be transferred to IR. As soon as instruction is completely in IR, the instruction mode flipflop is reset to EXECUTE state.

Execute state:- In this state, IAR transfers address to MAR and IOR transfers OP code to operand decoder. The decoder circuit decodes the OP code into subcommands which executes the instruction. One of the subcommand is a count pulse which steps the IC to next address. After execution of the instruction, the instruction mode flipflop is set to FETCH state to start new cycle.

Instruction Register (IR):- This register can be divided into two parts

(i) IAR Instruction Address register:- This 8-bit register contains the OP code of instruction currently being executed.

The four most significant bits of MBR are transferred to IAR.

(ii) IOR Instruction OP code register:- This 4-bit register transfers its contents to MAR and IC. It receives information from MBR.

This completes the system organization and specification of Educational Digital Computer (EDC). These specifications lead to the logical design in the next chapter.

## CHAPTER 3

### LOGICAL DESIGN

#### 3.1 Logical Symbolism and Diagrams:

Each wire in our logical diagrams is in one of two states (0,1) at each discrete moment of time. The two states 0 and 1 are complementary and they correspond to the truth values: false and true. Control wires are labelled as capital letters and information wires as small letters. Figure 3.1 shows the logical symbols and diagrams for building blocks.

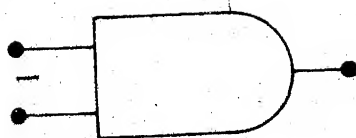
#### 3.2 Instruction Codes and Their Control Wires:

##### 3.2.1 Instructions and Subcommands:

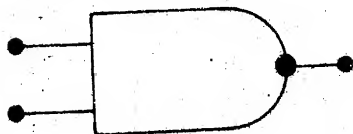
A description of machine instructions in terms of more elementary commands is being given to facilitate the design of arithmetic and control unit. Symbolic representation is used for the subcommands which is self explanatory.

- |           |   |
|-----------|---|
| 1) CLA n  | a) Transfer the contents of selected word to MBR<br>i.e., $\langle n \rangle \rightarrow \langle MBR \rangle$ |
|           | b) $\langle MBR \rangle \rightarrow \langle AC \rangle$   |
| 2) ADD n  | a) $\langle n \rangle \rightarrow \langle MBR \rangle$  |
|           | b) $\langle MBR \rangle + \langle AC \rangle \rightarrow \langle AC \rangle$                                  |
| 3) SUB n  | a) $\langle n \rangle \rightarrow \langle MBR \rangle$  |
|           | b) $\langle \overline{MBR} \rangle + \langle AC \rangle \rightarrow \langle AC \rangle$                       |
| 4) STO n  | a) $\langle AC \rangle \rightarrow \langle MBR \rangle$   |
|           | b) $\langle MBR \rangle \rightarrow \langle n \rangle$  |
| 5) LTRA n | $n \rightarrow \langle IC \rangle$  |
| 6) TRM n  | a) If $\langle AC \rangle$ -ve, $n \rightarrow \langle IC \rangle$  |
|           | b) If $\langle AC \rangle$ not -ve, $\langle IC \rangle + 1 \rightarrow \langle IC \rangle$                   |

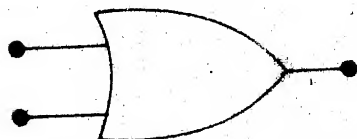
1 AND



2 NAND



3 OR



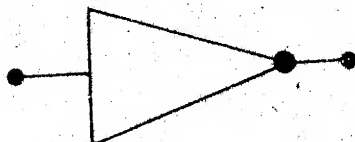
4 NOR



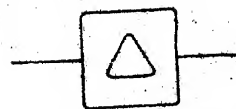
5 NON-INVERTING AMPLIFIER



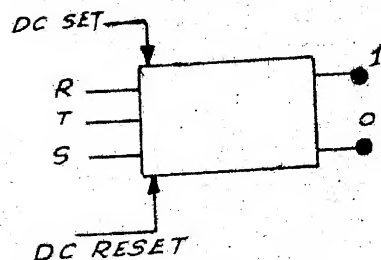
6 INVERTER



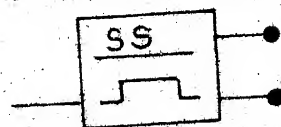
7 DELAY



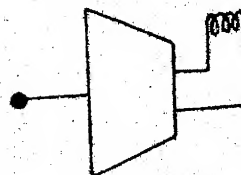
8 FLIP FLOP



9. MONOSTABLE



10 LAMP DRIVERS



1 NEGATION	$\bar{f}$	NOT $f$
2 AND	$f \cdot g$	$f$ AND $g$
3 OR	$f + g$	$f$ OR $g$
4 NOR	$\overline{f + g}$	NOT ( $f$ OR $g$ )
5 MATERIAL IMPLICATION ELEMENT	$f \supset g$	NOT $f$ OR $g$
6 EQUIVALENCE ELEMENT	$f \equiv g$	$f$ AND $g$ OR NOT $f$ AND NOT $g$
7. INEQUIVALENCE ELEMENT	$f \neq g$	$f$ AND NOT $g$ OR NOT $f$ AND $g$

FIG. 3-1. LOGIC SYMBOLS AND DIAGRAMS.

- 7) HLT                      Idle until computer is activated again manually.

### 3.2.2 Sequence of Events:

The general sequence of events in the operation of a single address computer is given below <sup>4,5</sup>

- 1)  $\langle IC \rangle \rightarrow \langle MAR \rangle$
- 2)  $\langle \langle MAR \rangle \rangle \rightarrow \langle MBR \rangle \rightarrow \langle IR \rangle$
- 3) What OP code?
  - a) CLA  $n$                $n \rightarrow \langle MAR \rangle$   
 $\langle \langle MAR \rangle \rangle \rightarrow \langle MBR \rangle$   
 $\langle MBR \rangle \rightarrow \langle AC \rangle$  , Go To 4
  - b) ADD  $n$                $n \rightarrow \langle MAR \rangle$   
 $\langle \langle MAR \rangle \rangle \rightarrow \langle MBR \rangle$   
 $\langle MBR \rangle + \langle AC \rangle \rightarrow \langle AC \rangle$  , GO TO 4
  - c) SUB  $n$                $n \rightarrow \langle MAR \rangle$   
 $\langle \langle MAR \rangle \rangle \rightarrow \langle MBR \rangle$   
 $\langle \overline{MBR} \rangle + \langle AC \rangle \rightarrow \langle AC \rangle$  , TO GO TO 4
  - d) STO  $n$                $n \rightarrow \langle MAR \rangle$   
 $\langle AC \rangle \rightarrow \langle n \rangle = \langle \langle MAR \rangle \rangle$  , GO TO 4
  - e) TRA  $n$                $\langle IAR \rangle \rightarrow \langle IC \rangle$  , GO TO 1
  - f) TRM  $n$               If AC is -ve,  $\langle IAR \rangle \rightarrow \langle IC \rangle$  , GO TO 1  
                          If AC not -ve,                      GO TO 4
  - g) HLT                      If start button is pressed, GO TO 4;  
                          else, stay in the same state
  - h) I/O                      Go To I/O control unit
- 4)  $\langle IC \rangle + 1 \rightarrow \langle IC \rangle$  , GO TO 1

State No. 1 & 2 are referred to as FETCH state, and 3 & 4 as EXECUTE state.

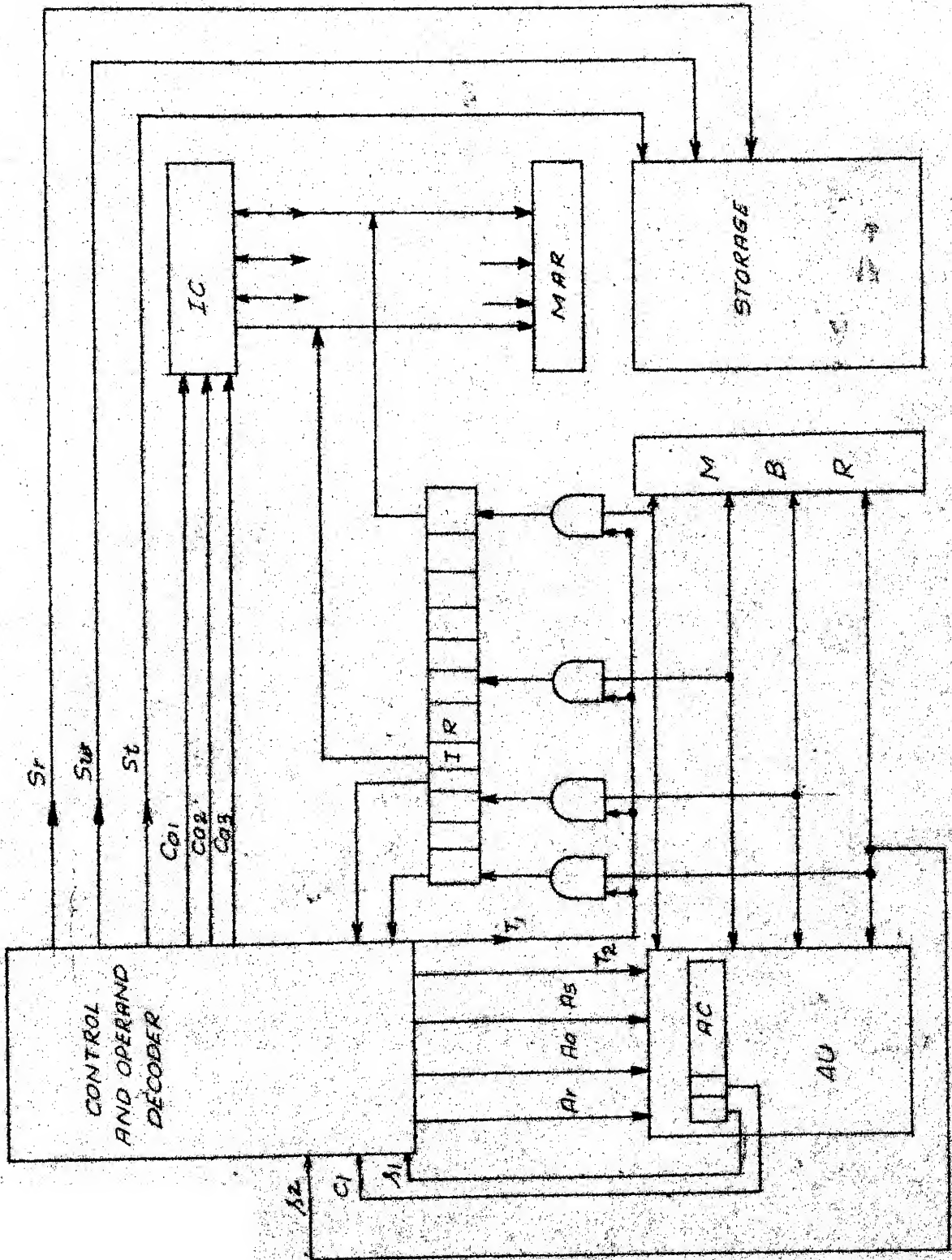


FIG. 3.2. INTERACTION BETWEEN DIFFERENT UNIT OF COMPUTER.

### 3.2.3 Elementary Subcommands:

If we go through the sequence of commands, we find that some of the commands are repeated in different instructions. Basic subcommands used in these instructions along with control wire nomenclatures are listed below

Subcommand	Nomenclature
1. $\langle I \rangle \rightarrow \langle MBR \rangle$	Sr
2. $\langle MBR \rangle \rightarrow \langle I \rangle$	Sw
3. To initiate the memory operation	St
4. $\langle MBR \rangle \rightarrow \langle IR \rangle$	T1
5. $\langle AC \rangle \rightarrow \langle MBR \rangle$	T2
6. $\langle IC \rangle \rightarrow \langle MAR \rangle$	Ca1
7. $\langle IAR \rangle \rightarrow \langle IC \rangle$	Ca2
8. $\langle IAR \rangle \rightarrow \langle MAR \rangle$	Ca3
9. $\langle MBR \rangle + \langle AC \rangle \rightarrow \langle AC \rangle$	Aa
10. $\langle \overline{MBR} \rangle + \langle AC \rangle \rightarrow \langle AC \rangle$	As
11. $\langle MBR \rangle \rightarrow \langle AC \rangle$	Ar
12. $\langle IC \rangle + 1 \quad \langle IC \rangle$	Cc

Interaction of different unit of the computer is shown in Figure 3.2, along with control and information wires. Control wires for I/O unit are discussed in section 3.5.

At any time  $t$ , four flipflops  $I1$ ,  $I2$ ,  $I3$  and  $I4$  are in exactly one of the sixteen distinct states. These distinct states together with instruction state flipflop and control clock completely determine the states of the twelve control wires labelled A, S, T and C with appropriate subscripts.

### 3.3 Arithmetic Unit:

#### 3.3.1 Arithmetic Operations:

a) Machine Addition:- Our task is to express

$$z = x + y \quad (1)$$

as a logical relation between  $x_i$ ,  $y_i$  and  $z_i$  ( $i=1,12$ ). We define  $C_{i-1}$  as the carry digit from the addition of three summands  $x_i$ ,  $y_i$  and  $C_i$ .

$C_{11}$  is 0 and  $z_i$  is 1 if and only if an odd number of the summands  $x_i$ ,  $y_i$  and  $C_i$  are 1. This is expressed by

$$z_i \equiv (x_i \oplus y_i \oplus C_i) \quad (2)$$

Since  $C_{i-1}$  is 1 just in case at least two of the three summands are 1, we have the equations

$$C_{i-1} \equiv (x_i y_i + x_i C_i + y_i C_i) \quad (3)$$

$$C_{11} \equiv 0 \quad (4)$$

b) Machine Subtraction:-

Machine subtraction of the word  $x$  from the word  $y$  can be defined in terms of following logical relations among bits  $x_i$ ,  $y_i$ ,  $C_i$  and  $z_i$ .

$$z_i \equiv (x_i \neq y_i \neq C_i) \quad (5)$$

$$C_{i-1} \equiv (\bar{x}_i y_i + \bar{x}_i C_i + y_i C_i) \text{ for } i > 0 \quad (6)$$

$$C_{11} = 1 \quad (7)$$

#### 3.3.2 Functions of the Arithmetic Unit:

In this subsection we consider three general functions performed by the arithmetic unit.

The first function is the transmission of information to the MBR. When instruction number 4 is executed, T2 will be stimulated (section 3.2),

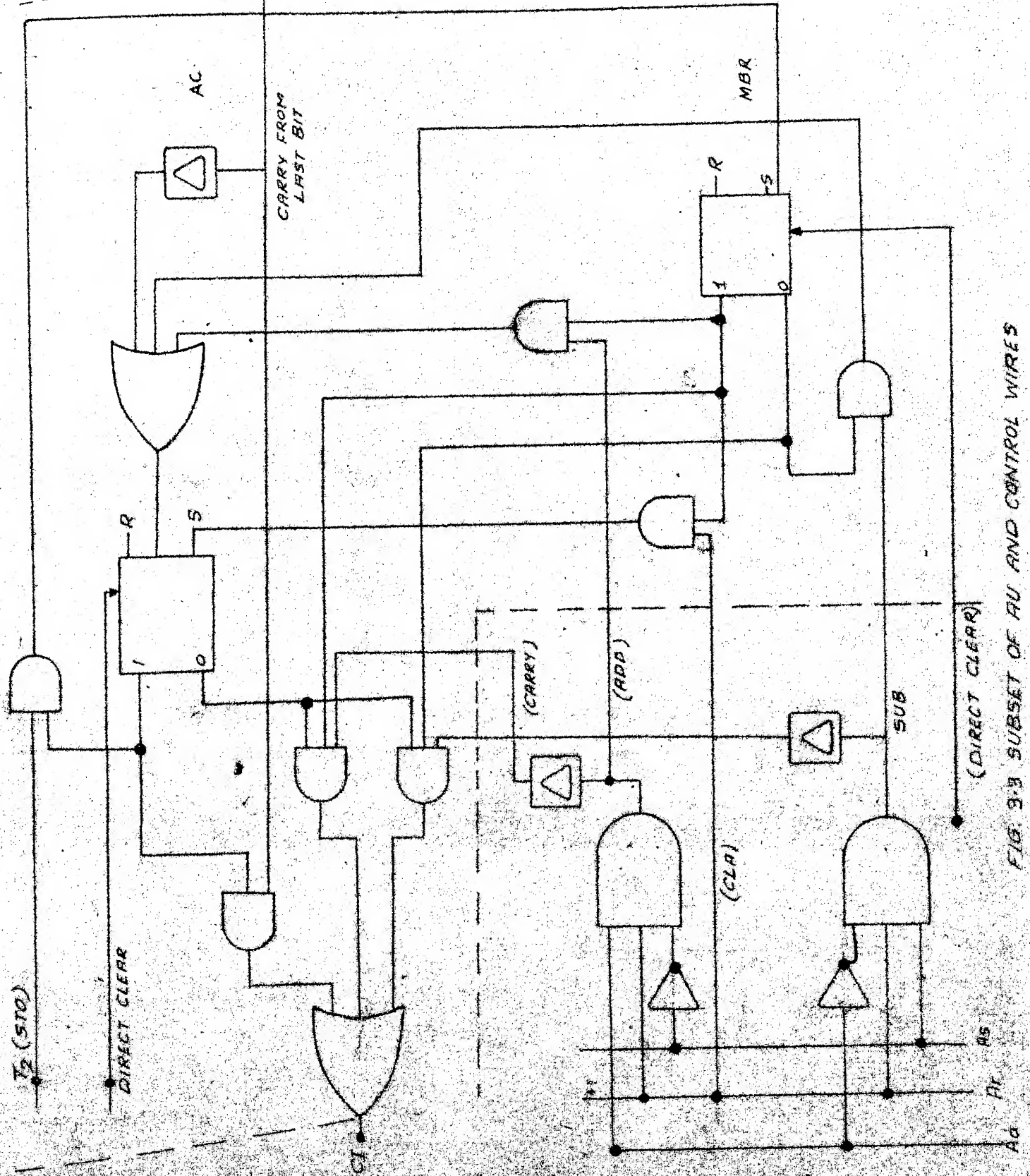


FIG. 3-3 SUBSET OF AU AND CONTROL WIRES

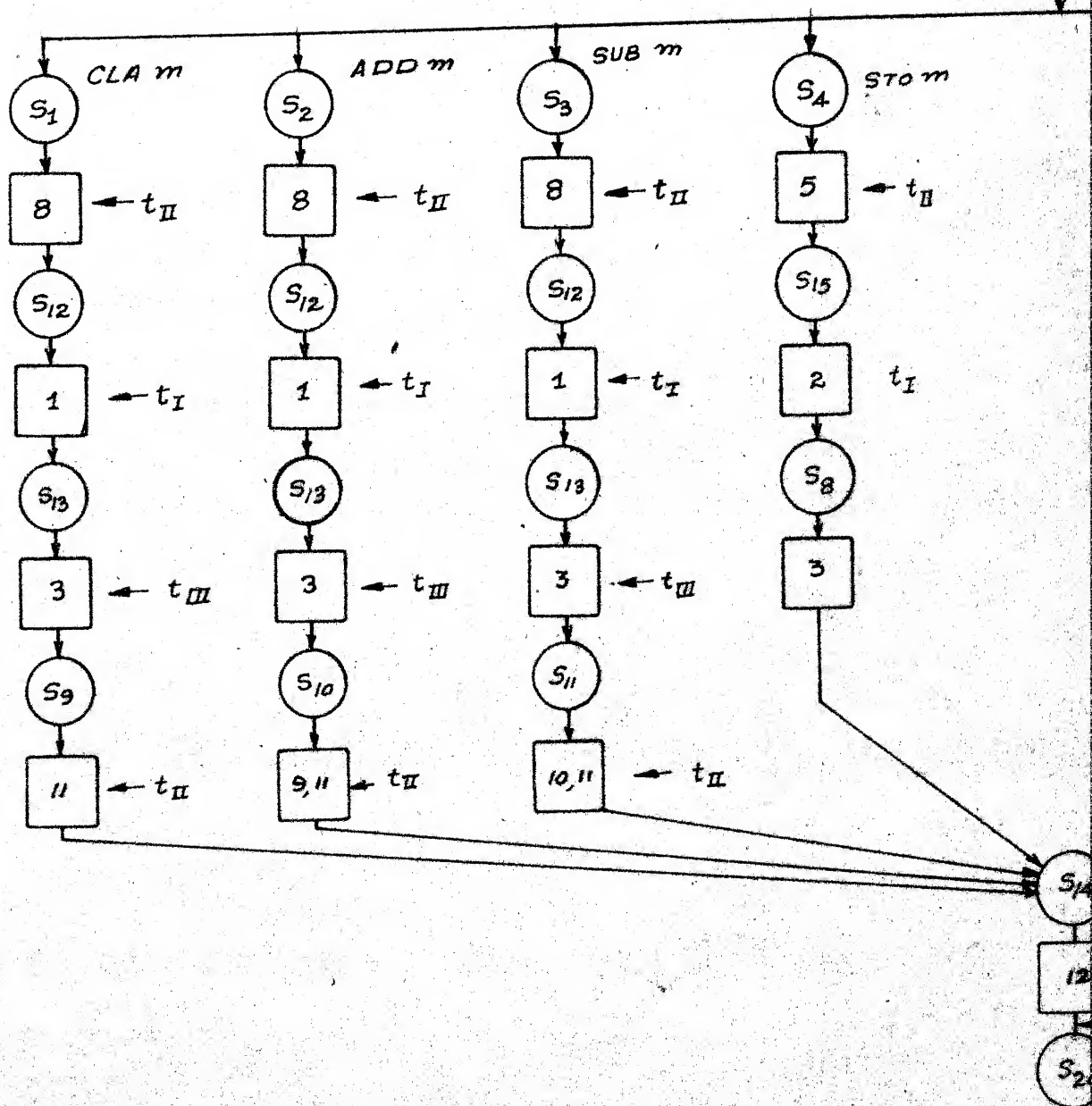
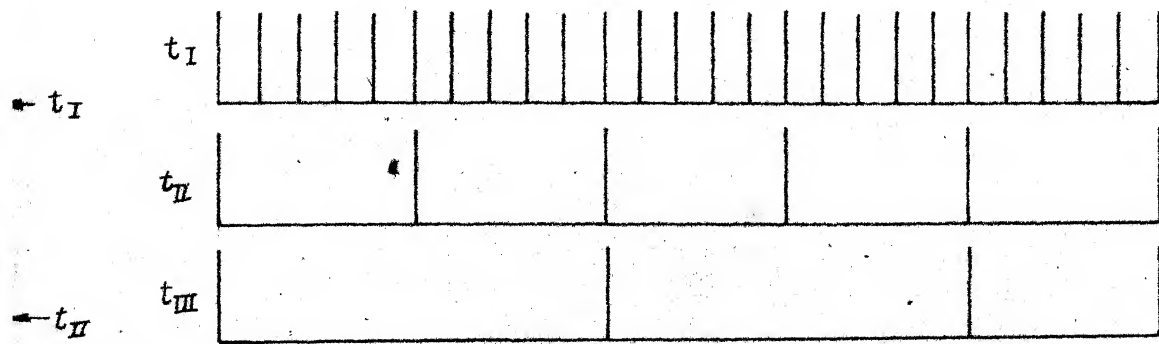


FIG. 3-4 FLOW DIAGRAM OF INSTRUCTION EXECUTION.

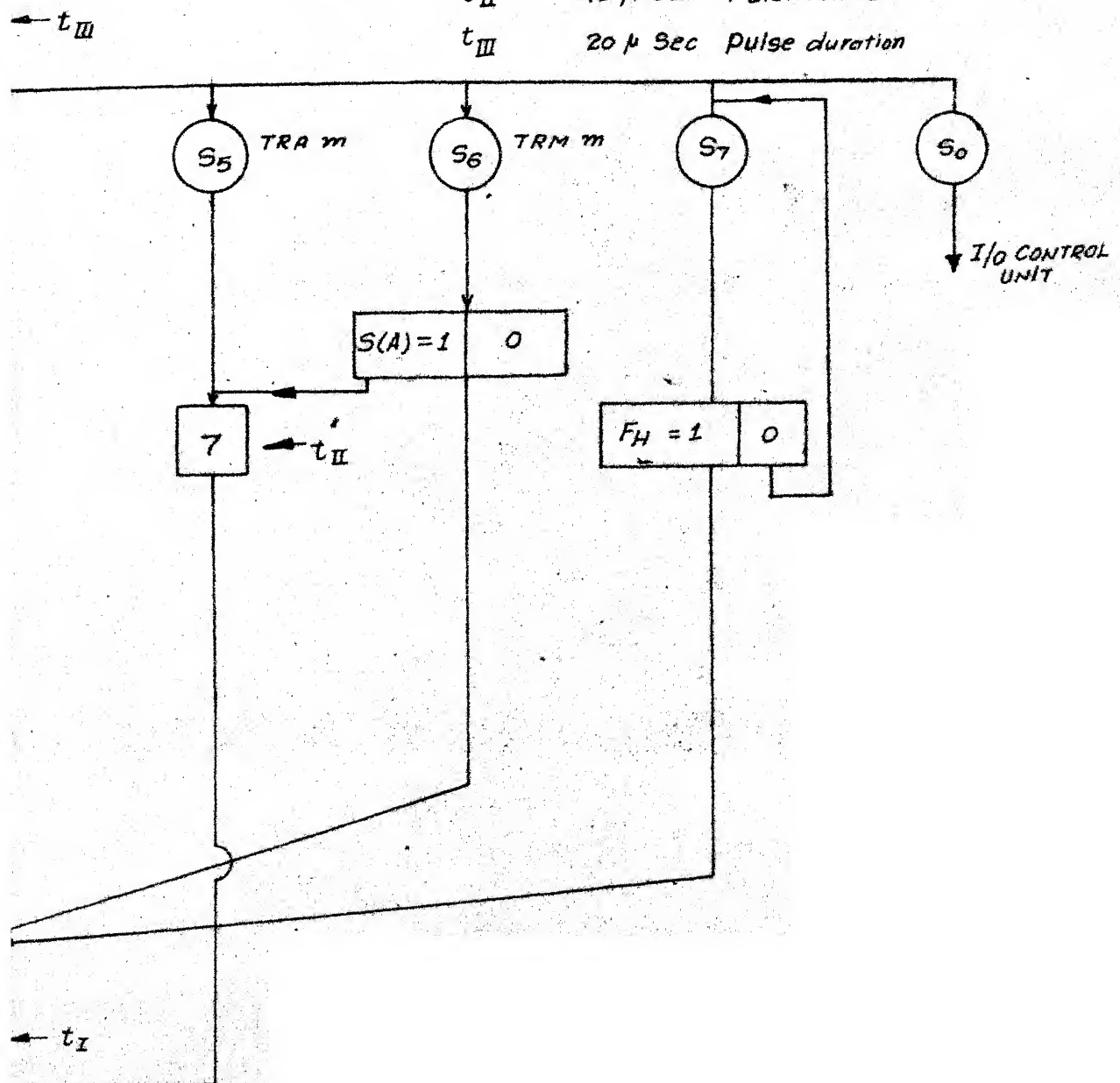


### CLOCK AND TIMING PULSES

$t_I$        $2 \mu \text{ Sec}$     clock

$t_{II}$        $10 \mu \text{ Sec}$     Pulse duration

$t_{III}$        $20 \mu \text{ Sec}$     Pulse duration



iii) CIA n:- For it to be executed the AU must realize following function

$$Ar \bar{As} \bar{Aa} \rightarrow \langle MBR \rangle \rightarrow \langle AC \rangle$$

### 3.4 Control Unit:

Control unit produces subcommand signals in proper sequence. Thirty two states of five flipflops (four IOR flipflops and one instruction state flipflop) determine the state of control unit. When command signal is activated CU is in one of these 32 states. Nineteen of these states are utilized for 12 command lines and seven op codes. States utilized for I/O unit are discussed in section 3.5. States for different OP codes are shown in figure 3.4.

A schenatic indicating successive states assumed by the control unit as well as different commands acitvated in the execution of each instruction is shown in Figure 3.5. The symbols in circle indicates state of CU and numbers in rectangles designate which commands are performed between the time interval of two states.  $t_I$ ,  $t_{II}$  and  $t_{III}$  are the three timing pulses.

$t_I$	clock pulse duration	2 $\mu$ sec.
$t_{II}$	Memory cycle	10 $\mu$ sec.
$t_{III}$	Machine cycle	20 $\mu$ sec.

The time required for the execution of each instruction, in terms of machine cycle is shown in Table 3.1.

#### 3.4.1 Instruction Counter:

Information transmission between instruction counter and other registers are shown in Figure 3.5(a). When subcommand 12 is acitivated (see section 3.2.3), contents of instruction counter is incremented by one.

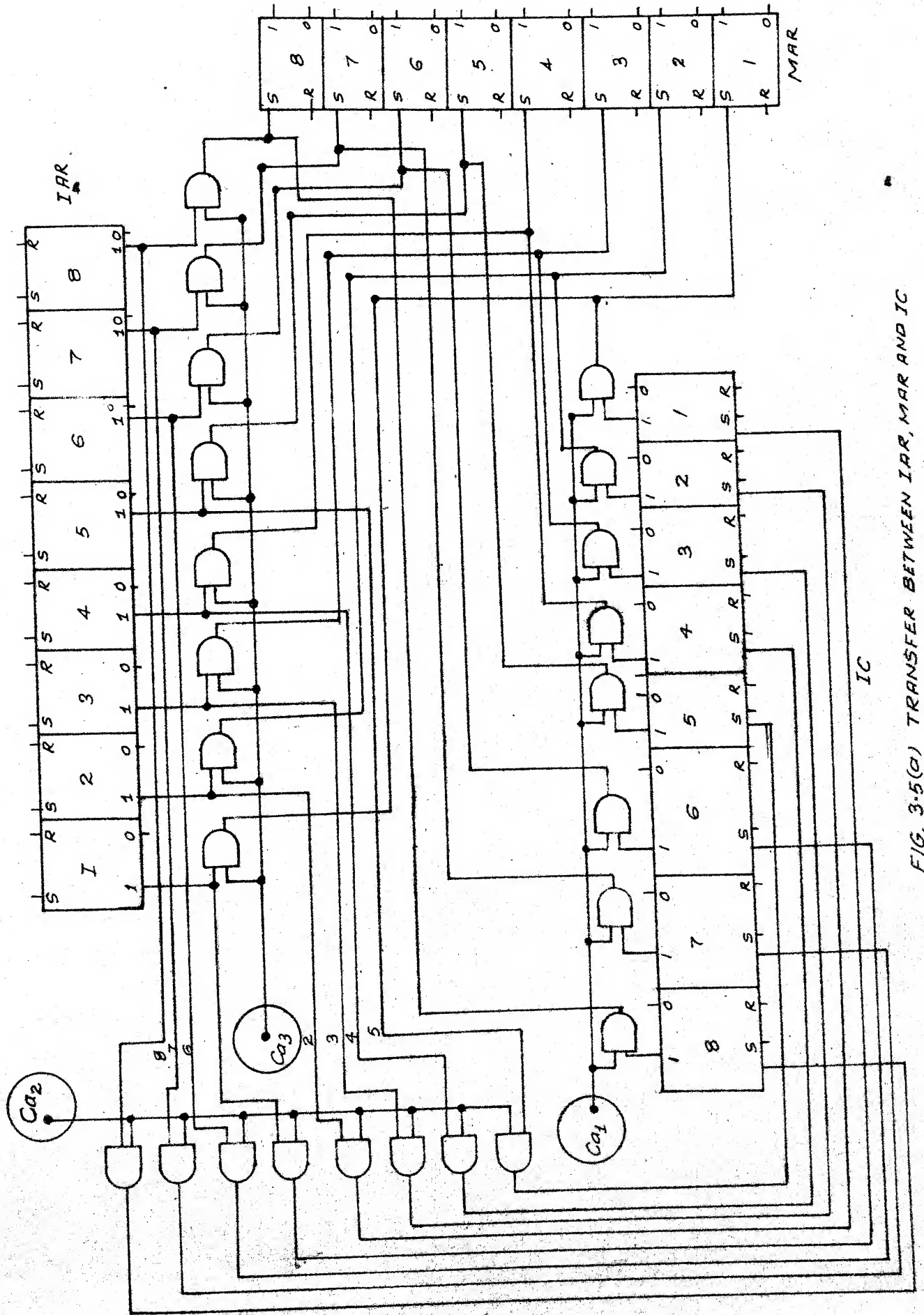


FIG. 3.5(G) TRANSFER BETWEEN IAR, MAR AND IC

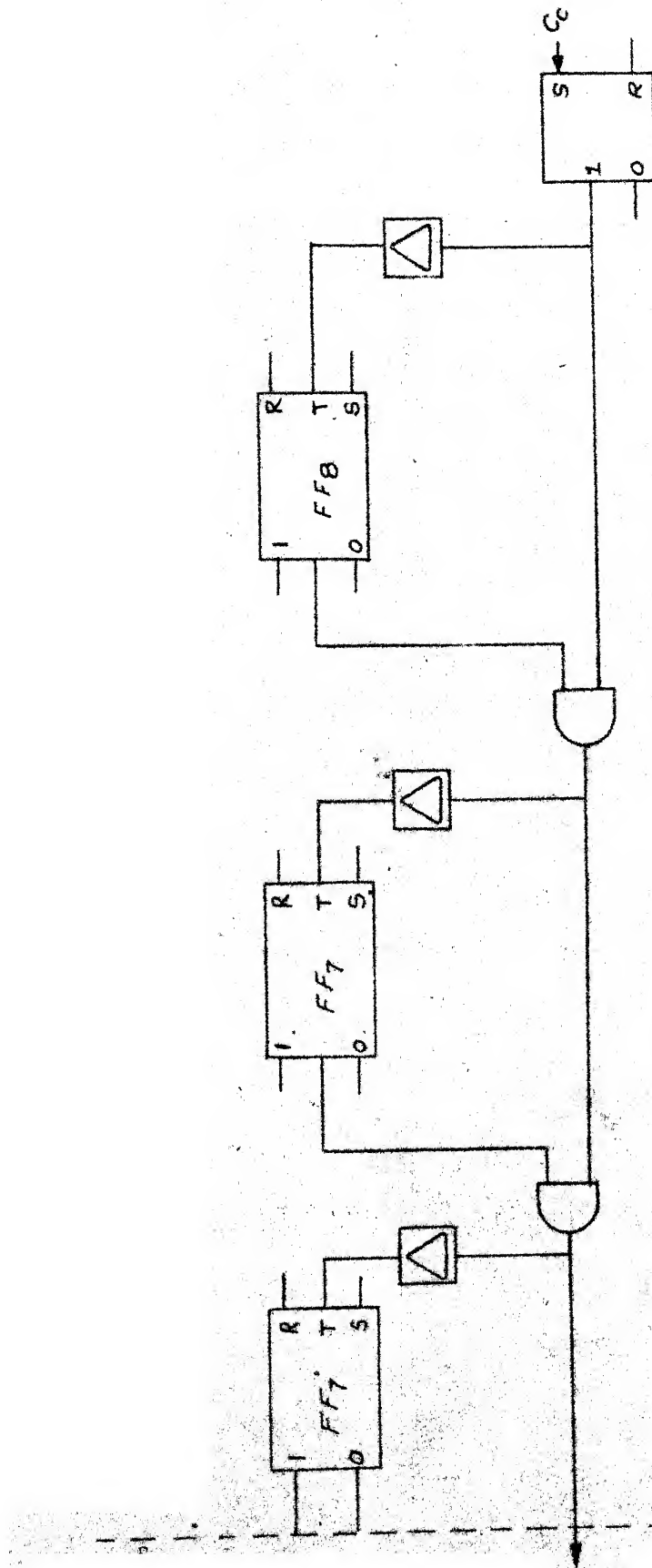


FIG. 3-5b ADDER WITH IC

TABLE No. 3.1

State assignment

No.	State	FF	I1	I2	I3	I4	
S0	0		0	0	0	0	
S1	0		0	0	1	0	CLA m
S2	0		0	1	0	0	ADD m
S3	0		0	1	1	0	Sub m
S4	0		1	0	0	0	STO m
S5	0		1	0	1	0	TRA m
S6	0		1	1	0	0	TRM m
S7	0		1	1	1	0	HLT
S8	0		0	0	0	1	
S9	0		0	0	1	1	
S10	0		0	1	0	1	
S11	0		0	1	1	1	
S12	0		1	0	0	1	
S13	0		1	0	1	1	
S14	0		1	1	0	1	
S15	0		1	1	1	1	
S16	1		0	0	0	0	
S17	1		0	0	1	0	
S18	1		0	1	0	0	
S19	1		0	1	1	0	
S20	1		1	0	0	0	
S21	1		1	0	1	0	
S22	1		1	1	0	0	
S23	1		1	1	1	0	
S24	1		0	0	0	1	
S25	1		0	0	1	1	
S26	1		0	1	0	1	
S27	1		0	1	1	1	
S28	1		1	0	0	1	
S29	1		1	0	1	1	
S30	1		1	1	0	1	
S31	1		1	1	1	1	

TABLE No. 3.3a

State assignment for I/O control unit

	I0	I1	I2	I4
SI1	0	0	0	1
SI2	0	0	1	1
SI3	0	1	0	1
SI4	0	1	1	1

TABLE No. 3.3b

Control wires for I/O Units

2	Octal	Output
3	BCD	Output
4	Octal	Input
5	BCD	Input
6	IOC1	$\langle AC \rangle \rightarrow \langle IOBR \rangle$
7	IOC2	$\langle IOBR \rangle \rightarrow \langle AC \rangle$

TABLE No. 3.2

Time taken by different Instructions in terms of m/c cycles

No.	Instruction	Fetch cycle	Execution cycle
1	CLA m	1	1
2	ADD m	1	1
3	SUB m	1	1
4	STG m	1	1
5	TRA m	1	-
6	TRM m	1	-
7	HLT	1	-

The following function must be realized by the instruction counter (see section 3.3.1) to perform its counting function properly.

$$\begin{aligned} z_8 &= \bar{x}_8 \\ C_{i-1} &= x_i \cdot C_i \\ z_i &= (x_i \neq C_i) \end{aligned} \quad i < 8$$

A binary adder is attached to IC to realize above function. (see Figure 3.5b).

Carry propagation time = 1.5 usec;  
for 8 bit IC

#### 3.4.2 Operand Decoder:

Every subcommand is a function of the state of the control unit. Commands are decoded from the four IOR flipflops and instruction state flipflop. Decoding is completed from the state assignment (Table 3.1) and their sequence from the instruction flow diagram (Figure 3.4). Input to the IAR flipflops and instruction state flipflops is function of its present output and control clock (Figure 3.4).

#### 3.4.3 Control Clock:

Basic clock in the system is of 2  $\mu$ sec. Two more timing pulses of 10  $\mu$ sec. and 20  $\mu$ sec. are generated by counting it down. 10  $\mu$ sec. pulses are required for memory operation and 20  $\mu$ sec pulses for basic machine cycle.

#### 3.4.4 Overflow detection:

Overflow can occur in 2's complement arithmetic in two ways:

- 1) Augend, addend both +ve and there is carry from bit position 1

$$\text{i.e. } \bar{S}_1 \bar{S}_2 C_1 = 0v$$

$$S_1 = \text{Sign of Augend}$$

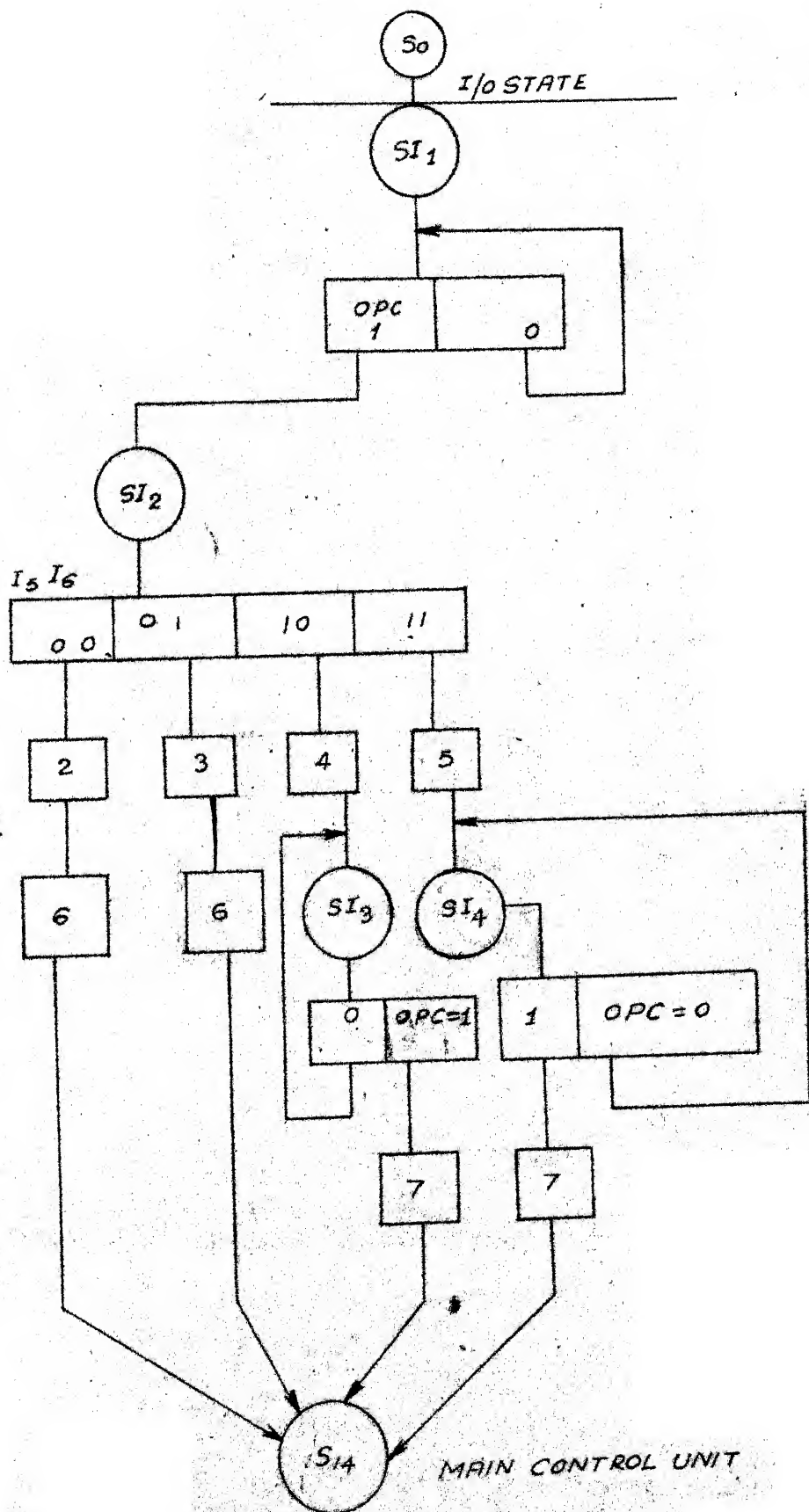


FIG. 3-6 FLOW DIAGRAM OF I/O CONTROL UNIT.

$S_2 =$  Sign of Addend

$C_1 =$  Carry from bit position 1

2) Augend and addend both -ve and there is no carry from bit position 1.

$$\text{i.e., } s \quad S_1 S_2 \bar{C}_1 = 0v$$

So overflow detection circuit must realize following function

$$\bar{S}_1 \bar{S}_2 C_1 + S_1 S_2 \bar{C}_1 = 0v$$

### 3.5 Input-Output Control Unit:

I/O control unit produces sequence of subcommand signals to transfer the information between AC and IOBR (Input-Output Buffer Register). I/O instruction format is given in Sec 2.2 and sec 2.3. When state  $S_0$  is encountered in the regular operation of control unit (Figure 3.4), control is transferred to I/O microprogramme unit.

A schematic indicating successive states assumed by the flipflops  $I_0$ ,  $I_1$ ,  $I_2$  and  $I_4$  and as well as different subcommands activated in the execution of I/O unit is shown in Figure 3.6.

I/O control unit monitors the operation complete line (OPC) and if it is busy (OPC=0) it continues to stay in state SI1 (Table 3.3). If it finds OPC line free (OPC=1) it activates one of the subcommands 2, 3, 4 and 5 depending upon the state of flipflop  $I_5$  and  $I_6$  (Table 3.4). If output line is energized (command 2 or 3) contents of AC is transferred to IOBR. by command 6 (IOC1). In case of input line (command 4 or 5), I/O control unit goes to state SI4 and SI5. OPC line is again monitored and if it is not free, I/O CU continues to stay in this state. As soon as OPC line becomes free subcommand 7 is energized which transfers content

of IOBR to AC and control is transferred to the state S14 in main control unit and normal sequence is continued.

### 3.6 The Console:

The console (Figure 3.7) provides indicators and controls for the performance of following functions.

- 1) Power Control
- 2) Operation Control
- 3) Program test
- 4) Manual input
- 5) Register Display

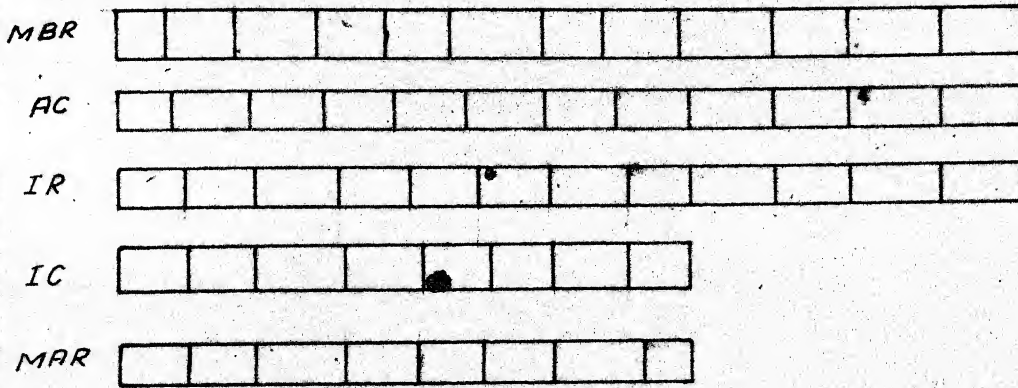
The logic associated with each of these functions is discussed briefly.

- 1) Power Control:- In this group are included AC power dc power switches and fuses.
- 2) Operation Control:- In this group are included the RESET, START, STOP and mode switches. Indicators in this group are operate, teleprinter Input ON, teleprinter output ON and mode of input and output (Octal or BCD).

The RESET button resets all the flipflops and clears the memory. The START and STOP switches control machine sequencing. The START push button sets halting flipflop  $F_H$  (see Figure 3.4) to 1 and thus control the increment of IC. As long as  $F_H$  is set to 1 START lamp is lit. The STOP push button inhibits the function of IC i.e. sequencing stops.

The mode selection group has three push button switches. Automatic, Manual and Load. In Automatic mode normal sequencing occurs. In the Manual mode input switches are made operative. In Load mode a particular

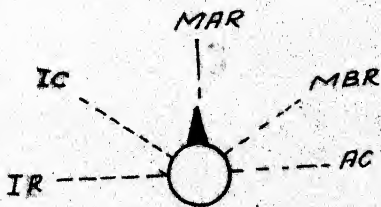
## REGISTER DISPLAY



## INPUT SWITCHES



## REGISTER INPUT SELECTOR



## PUSH BUTTON



HLT

PARITY  
CHECK

OVER FLOW



RESET

START

STOP

AUTOMATIC

MANUAL

LOAD

INPUT

OUTPUT



STATUS OF I/O



TELEPRINTER I/O

## POWER SWITCHES

AC POWER  
SWITCHDC POWER  
SWITCH

## FUSES



FIG. 3.7. CONSOLE OF EDC

location number is introduced in the IC and then automatic sequencing begins.

Five indicators are provided for teleprinter I/O. INPUT-OCTAL, INPUT-BCD, OUTPUT-OCTAL, OUTPUT-BCD and I/O operation complete indicator (OPC) is provided to indicate whether teleprinter is busy or not. If OPC indicator is ON, teleprinter is free otherwise it is busy.

- 3) Programme Test:- Halt, Parity error, and Overflow indicators are provided in this group.
- 4) Manual Input:- Manual input is controlled by the input bit switches and the register input selector. Bits are entered in by placing the input bit switches in ON or OFF position. The setting of the register input selector determines to which register this number is transferred. The actual shifting of the bits into the chosen register starts when the push button is depressed.
- 5) Register Display:- The register indicator lamps are connected to the outputs of each of the flipflops composing the registers. An indicator is On when one is stored, OFF when zero is stored.

## CHOICE OF LOGIC FAMILY

**4.1 Different Logic circuit configuration:**

There can be no single logic circuit configuration best suited for all applications. The selection of a circuit type depends on the particular situation and the compromises and trade off's permissible for the application concerned. An attempt is made here to select logic family best suited to Indian manufactured components. Some of the important logic circuits are compared (Table 4.1) keeping into account the following characteristics

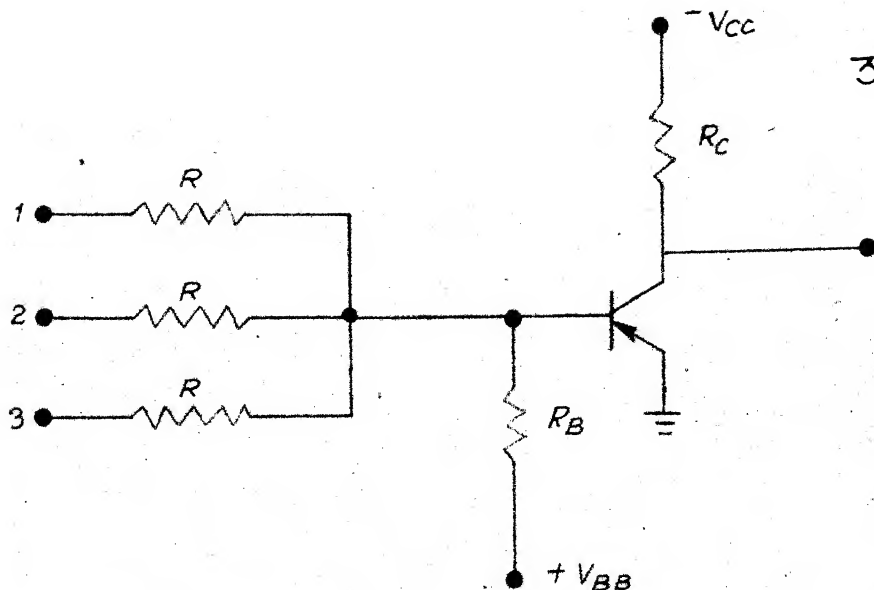
1. speed (propagation time delay)
2. Noise immunity
3. Fan-in and Fanout capabilities
4. Power supply requirements
5. Power density shown when packaged
6. Cost

On the basis of comparison made in Table 4.1 three logic families RTL, DTL and TTL are discussed in detail (Fig. 4.1)<sup>6</sup>

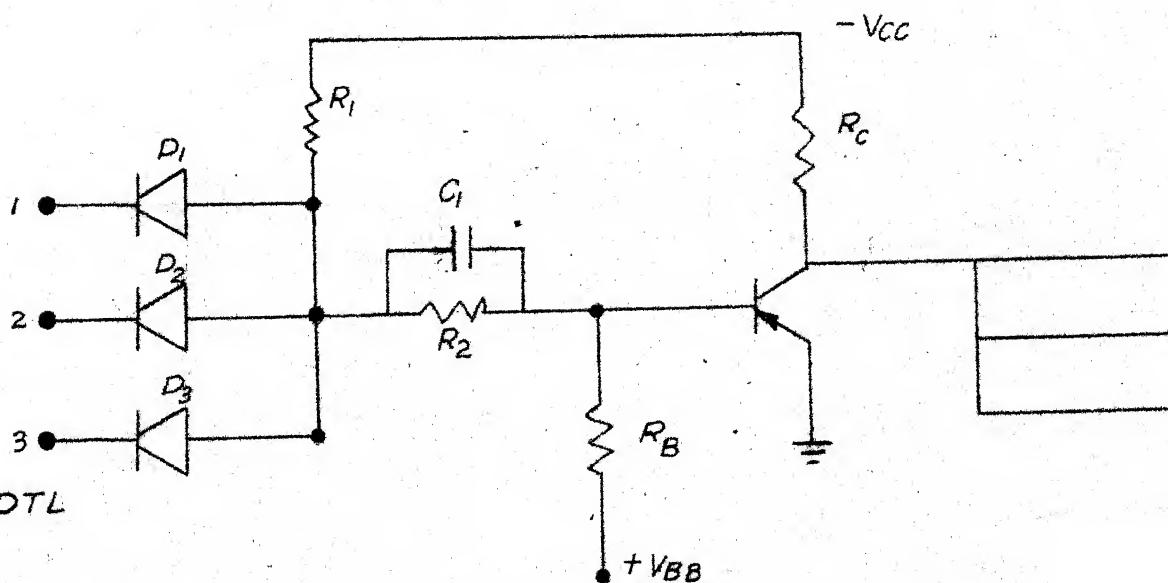
**4.2 RTL, DTL and TTL Logic circuits:**

RTL: For low speed operation RTL family is more suitable. It is one of the cheapest logic family. Power supply requirement is same as DTL circuits. There is no isolation between different input channels and this increases cross talk in RTL circuits. Some of the desirable properties of transistor in RTL application is listed below:

(a) RTL



(b) DTL



(c) TTL

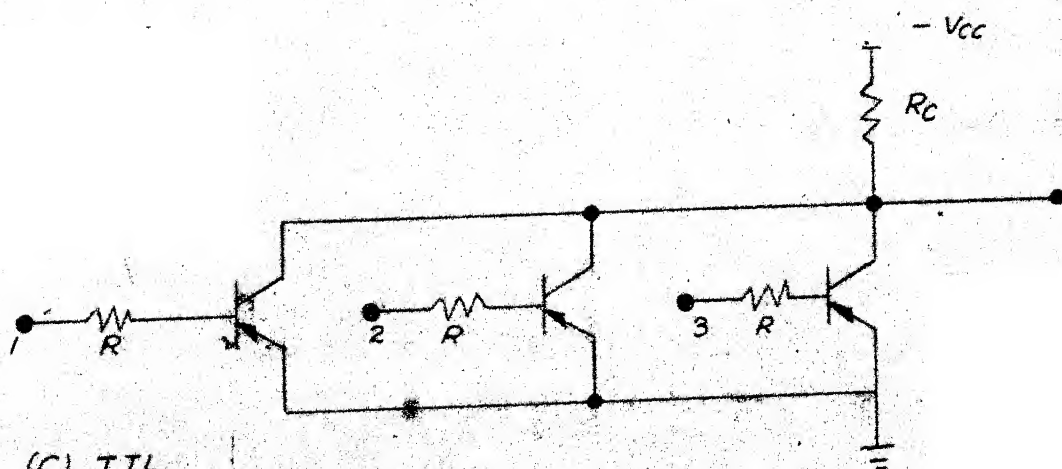


FIG. 4-1. LOGIC CIRCUIT CONFIGURATION.

## Comparison of Different Logic Circuits

No.	Logic Family	Propagation Speed	Power Requirements.	Noise Immunity	Fan out	Cost
1.	DCTL	4	4	1	2	moderate
2.	RTL	2	2	2	2	less
3.	DEL	2	3	4	2	moderate
4.	ILL	2	3	4	2	moderate
5.	TTL	4	4	2	4	high

1 bad

2 fair

3 good

4 excellent

TABLE 4.2a

DC Beta as a function of fan in and fan out

$$V_C = 12v$$

$\begin{array}{c} M \\ \diagdown \\ N \end{array}$	1	2	3	4	5
1	3	6	9	13	15
2	4	8	12	16	20
3	5	10	15	20	25
4	7	14	21	28	35
5	11	22	33	44	55

TABLE

DC Beta as a function of fan in and fan out

$$V_C = 6 V$$

$\begin{array}{c} M \\ \diagdown \\ N \end{array}$	1	2	3	4
1	3	6	9	12
2	5	10	15	20
3	14	28	42	56
4	$\infty$			

1. Low  $V_{CESAT}$

2. High  $V_{BEON}$

3. Low  $I_{CO}$

4. High beta

7  
 Dc beta as a function of fan-in and fan-out is given in Table 4.2a and Table 4.2b

Following design results are obtained from Table 4.2a and 4.2b

- 1) RTL with higher  $V_C$  is less costlier than low  $V_C$  circuits.
- 2) Transistors can be grouped in high and low beta group.

This will help in utilizing better transistor for high fan-in and fan-out application, transistor with lower beta can be used in inverter circuits.

DTL: DTL family does not provide any specific advantage over RTL, except that cross-talk between input channel is minimized and power dissipation is less. One of the major drawback of DTL circuit is its limitation of fan-in.

It has been found in the existing market prices that switching diodes are almost in same price range as transistors and so DTL family is much costlier than RTL.

TTL: TTL offers following advantages in comparison to RTL and DTL.

- 1) Principal advantage of TTL is its low power and high speed operation.
- 2) Although it uses more transistor, the reduction in the number of other components such as diodes and resistors, is a significant advantage.

3) This family is not much costlier than DTL.

4) Only one power supply is used and this could be a very prominent factor in some application.

TTL family is chosen for high speed operation and building blocks for EDC has been made with this logic.<sup>8</sup> Blocks have been built and tested under extreme conditions. Separate set of building blocks using RTL logic has been designed for low speed operation.<sup>9</sup>

## DIGITAL LOGIC SIMULATION

5.1 Need of Logical Check:

There are two fundamental problems faced by the designer of electronic digital system.<sup>10</sup>

- i) Synthesis:- Given a description of the required function of a device, implement a minimal circuit which performs the function.
- ii) Analysis:- Having designed a circuit he would like to analyze it for its performance and also determine associated factors like reliability with respect to component tolerances and economical hardware realization.

It is very common to make minor errors in synthesis which will lead to malfunction of the system. There is considerable variation in delays associated with each circuit blocks. It is precisely these delays that create logical hazards in digital circuits. Elimination of these hazards and their detection is one of the main problem of digital system design. The designer must find out which subsystem is critical from the point of view of hazards.

It is therefore necessary to carry out exhaustive testing of the system before producing a prototype for final fabrication. This chapter aims to describe a technique of digital logic simulation through a computer program written in Fortran IV language to analyze a system without too much trouble.

## 5.2 Program Characteristics:

The user can supply the details of the block diagram in a convenient form. The circuit can be tested digitally for any specified input waveforms and the output of desired blocks are displayed in a manner similar to an oscillogram. Races, hazards and improper logic can be easily detected by visual observation almost like one would do on a bread board without any hardware troubles.

## 5.3 Simulation Technique:

### 5.3.1 Circuit blocks to be modelled:

One guide line used in the selection of the subblocks of the model is that, the set should make it possible to deal with almost all digital systems. It is assumed that the circuits to be analyzed are built entirely from the blocks given below:

1. AND :      $M$  ( $M \leq 5$ ) input AND block with one output  
 $ND_{\max} = 36$  unit of time  
 The unit of time will be a convenient fraction of clock time.
2. OR         $M$  ( $M \leq 5$ ) input OR block with one output  
 $MD_{\max} = 36$  unit of time
3. NOT       One input NOT block with one output  
 $ND_{\max} = 36$  unit of time
4. NOR        $M$  ( $M \leq 5$ ) input NOR block with one output  
 $ND_{\max} = 36$  unit of time.
5. FLIPFLOP Five input flipflop block with one output  
 $ND_{\max} = 36$  unit of time  
 Five inputs   DC   RESET  
                   DC   SET  
                   PULSE SET  
                   PULSE RESET  
                   TRIGGER

Edge  $0 \rightarrow 1$  or  $1 \rightarrow 0$  brings out action in the case of pulse set, reset and trigger input. Complementary is not available.

6. DELAY One input and one output Delay block  
 $ND_{\max} = 360$  unit of time

7. MONOSTABLE One input and one output MONO block

Time duration of monostable

$ND_{\max} = 360$  unit of time

Either of the two edges  $0 \rightarrow 1$  and  $1 \rightarrow 0$  can trigger monostable.

8. INPUT Input to the system is treated as a separate circuit block - Input generator

No input terminal, one output.

It provides input to the system at prescribed time and sequence. String length should be multiple of word length of computer.

INPUT WORD (INPWD = 10 words) = 360 unit of time

Delays associated with each block ( $ND_{\max}$ ) and number of inputs to a gate (M) can be increased very easily.

### 5.3.2 Circuit Structure:

#### 1) Circuit List:

Circuit structure is represented by specially formed lists called circuit lists. Every blocks are associated with a block list:

$$BL(K) = \langle K, NT(K), ND(K), (NI(K, J), J=1, M) \rangle$$

where K = block number

$NT(k)$  = Type of block

$ND(k)$  = Delay associated with block K.

$NI(K, J)$  = Input to the block K

Circuit list is formed by using block lists  $BL(K)$  as given below

```

CIRCUIT      Head of the list
BL(K1)
BL(K2)
.
.
.
BL(Kmax)     CEND

```

CEND is used to terminate the list.

## 2. INPUT list:

A separate list INPGEN(K) is associated with each of the input generator

$$\text{INPGEN}(K) = \langle K, \text{NT}(K), \text{ND}(K), \text{FBSEQ} \rangle$$

i1, i2, i3, i4, .....

where FBSEQ denotes whether first input bit is 0 or 1. If

FBSEQ is FALSE, sequence of input is as shown below

```

i1  i2  i3  i4  .....
0   1   0   1

```

and if FBSEQ is TRUE, 0 and 1 are complemented. Input list is formed by using INPGEN(K) lists as shown:

```

INPUT      Head of the list
INPGEN(K1)
INPGEN(K2)
.
.
.
INPGEN(Kmax)  END

```

where K1, K2 ..are block numbers, Kmax is the maximum number of input blocks. END is used to terminate the list.

## 3. OUTPUT LIST:

In addition to the inputs of the circuit output points are specified as output list.

OUTPUT

K1

K2

.

.

.

Kr

Head of the list

ZERO

where K1, K2 .. are block numbers. ZERO is used to terminate the list. Both type of logic (-ve or +ve) can be used for plotting the waveform.

#### 4. Initial state of the blocks:

There are two modes of initial state specification.

- a) Normal Mode:- Initial output of all the blocks and contents of all the delay words are taken as 0.
- b) RESTART - START mode:- The analysis of system can be carried out in steps by using Restart-Start mode of initial state specification. If restart option is used, output of all the blocks and delay blocks will be punched in binary form when the program is terminated. These punched results may be used as the starting condition for next set of input string by just using START option. This mode of operation is very useful when system has to be checked for longer duration of time than what the input block allows.

#### 5.3.3 Evaluation:

The output is updated at every unit time (fraction of clock time). Temporary output of all the blocks, ZT, are determined from logical function. Interim output  $y(K)$  is obtained from delay word associated with the block. ZT is inserted in the delay word and then the whole block of delay word is shifted in left direction, by one bit. With every

block one logical variable (PROC) is associated which is made TRUE when block is processed. After processing all the blocks  $y(K)$  is transferred to  $z(K)$  before start of next clock pulse.

46

#### 5.4 Computer Program Structure:

The entire program is written in FORTRAN IV language (see Appendix 3). System flow chart is given in Appendix 2. A reference manual is provided in Appendix 1 to help the user to prepare data and certain control cards.

No error diagnostic routine is provided to locate errors in preparing different tests. Data is printed along with the results. In the case of monostable block a warning message is printed if the monostable is triggered before the circuit has completed a cycle. Input pulse to monostable is ignored. Content of input word is printed in Octal form for checking input packing routine. Results of all the intermediate operations and contents of each delay words can be printed by giving TRUE value to logical variable SWITCH.

Results of some sample problems are given in Appendix 4 to illustrate the use of the program.

## CHAPTER 6

### CONCLUSION

The design of the system EDC provides guide line for the construction of the small computer. Almost all the components required by this computer can be obtained in India and its construction is possible with hundred percent indigenous components. The TTL logic configuration, chosen for this computer is the best logic family for the design of digital systems in 500 Kc/s speed range.

The design carried out in previous chapters also provides takeoff points for the further development of the system. Memory capacity of the computer can be increased to 1K with the available cores. One spare bit provided in instruction OP code register (IOR) can be utilized to increase the number of instructions.

The logic simulator program provides a simple method for testing the logical design of any digital system. The outputs of the desired block can be displayed in a manner similar to an oscillogram. Races, hazards and improper logic can be detected by visual observation of the waveform. At present the program does not provide any dynamic method to change the delays associated with blocks during the simulation. A delay change routine can be incorporated to do this. This helps the designed in isolating those sections of the system when slower circuits can be used and thereby reduce the total cost.

### References

1. Braun, E.L., "Digital Computer Design"  
Academic Press, Newyork 1963
2. Sondhi, V., "Design of a coincident current memory system"  
M.Tech. Thesis (EE 14-68) IIT, Kanpur
3. Krishnaswami, G., "Teleprinter Input-Output to a small Computer"  
M.Tech. Thesis (EE-18,68) IIT, Kanpur
4. Phister, M.Jr., "Logical Design of Digital Computer", John Wiley,  
1960
5. Burks, A.W.; Cppi, I.M., "The Logical Design of an Idealized  
General purpose Computer"  
J. Of Franklin Institute, vol.251, 1956  
pp 299 - 314, pp 421-436
6. Khambata, A.J., "Introduction to semiconductor Integrated Circuits"
7. Stasior, R.A., "Application note", General Electric 90-22, 12/64
8. Lt. Majumdar, S., "1 Mc/s Universal Logic Modules"  
M.Tech. Thesis (EE-15-68) IIT/Kanpur
9. Bhat, M.V., "Design of Electronic Desk Calculator"  
M.Tech. Thesis (EE-13-68) IIT, Kanpur
10. Ieon Shalla, "Automatic Analysis of Electronic Digital Circuits"  
CACM, vol 9, No. 5, May 66.

## APPENDIX

# APPENDIX I

## REFERENCE MANUAL

### Data Preparation

#### Data Preparation

The data cards are punched as given below:

#### Card Number

1. LOGIC, START, RESET, EDGE, SWITCH  
(5(L6,6X))
2. KIMAX, NOMAX, KIMAX, INVM, INPWD, INPTM, KIDMX  
(7(I3,4X))
3. Output Format  
(1X,I3,3X,i(A4,2X),6X,r(A4,2X))
4. Title  
(1X,nH INPUT OUTPUT )
5. Input and output block numbers  
(nX,i(I3,2X),6X,r(I3,2X))

where i= no. of input blocks

r= no. of output blocks

n= no. of blanks required

6. OUTPUT

(A6)

7. K1

8. K2

. .

. .

nc0 Kr

ZERO

```

      (I3,71X,A6)

      nco= NOMAX+7

nco+1  CIRCIT

      (A6)

nco+2  K, NT(K), ND( K), (NI(K,I),I=1,M)
      .
      .
      .
      .

ncc                                CEND

      (I3,2X,A6,2X,M(I3,1X),nX,A6)

      where n should be so adjusted that CEND starts from
      column 75.

      ncc = nco+(KMAX-KIMAX)

ncc+1  K,NT(K), ND(K), FBSEQ

      i1  i2  i3  .....

ncc+2  .
      .
      .
      .

ncmax  .                                END

      (I3,2X,A6,I3,2X,L6,56X,A6)

```

where i's are punched in 3 column fields, right adjusted and may continue to any number of cards.

Multi delay word type of blocks (IND,DELAY,MONO) are numbered first starting from 2, followed by other blocks. 0 and 1 are reserved for false and true levels and floating inputs are represented by -1. (See Sample problem in the Appendix 4)

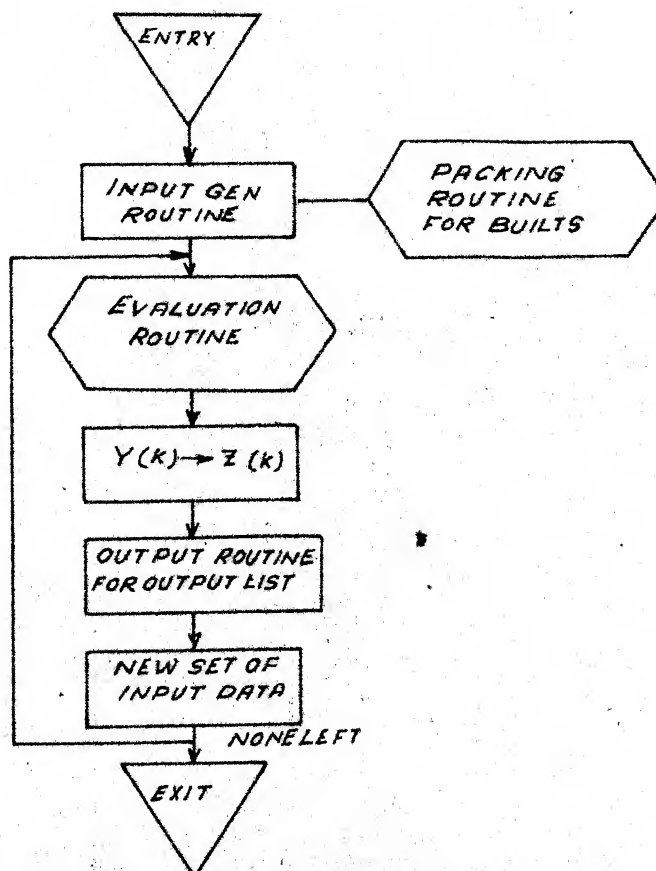
APPENDIX 2

FIG. 5.1(a) MAIN PROGRAM.

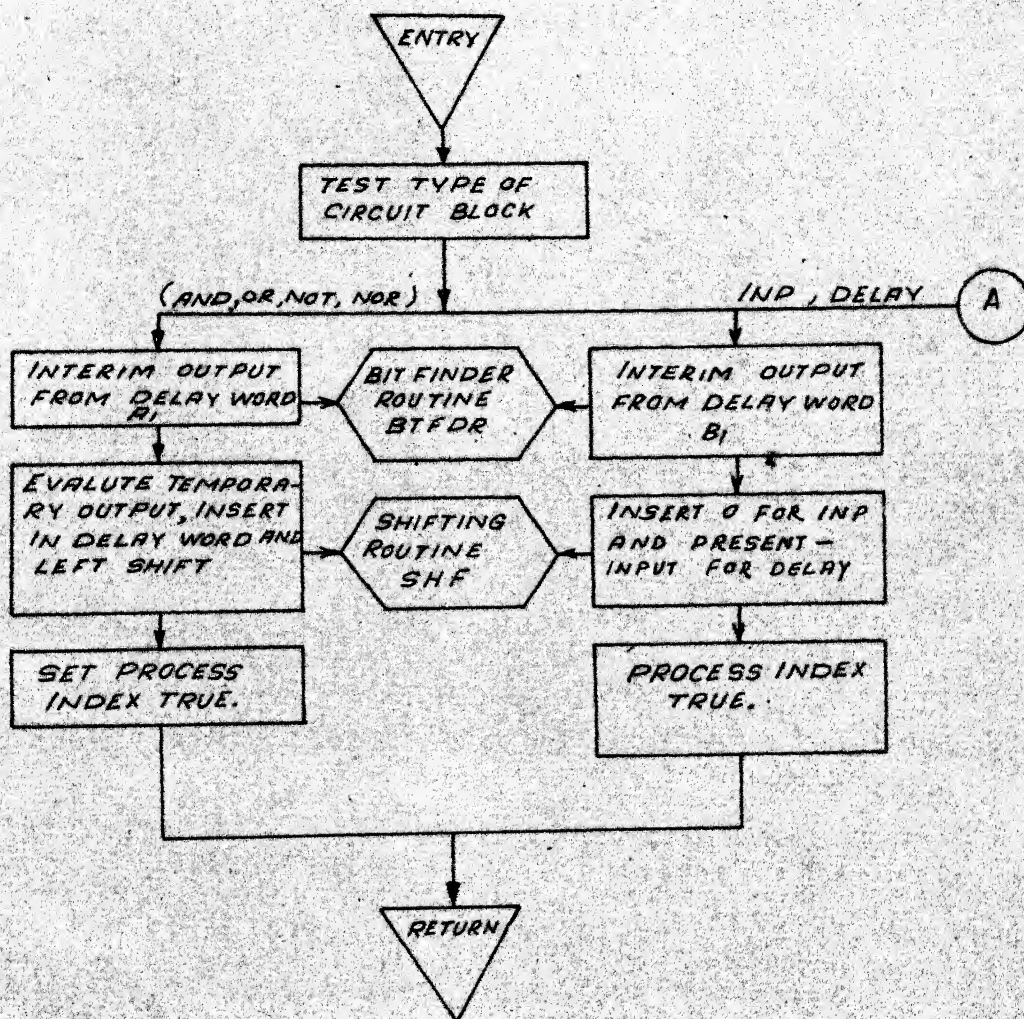


FIG. 5.1(b) EVALUATION ROUTINE

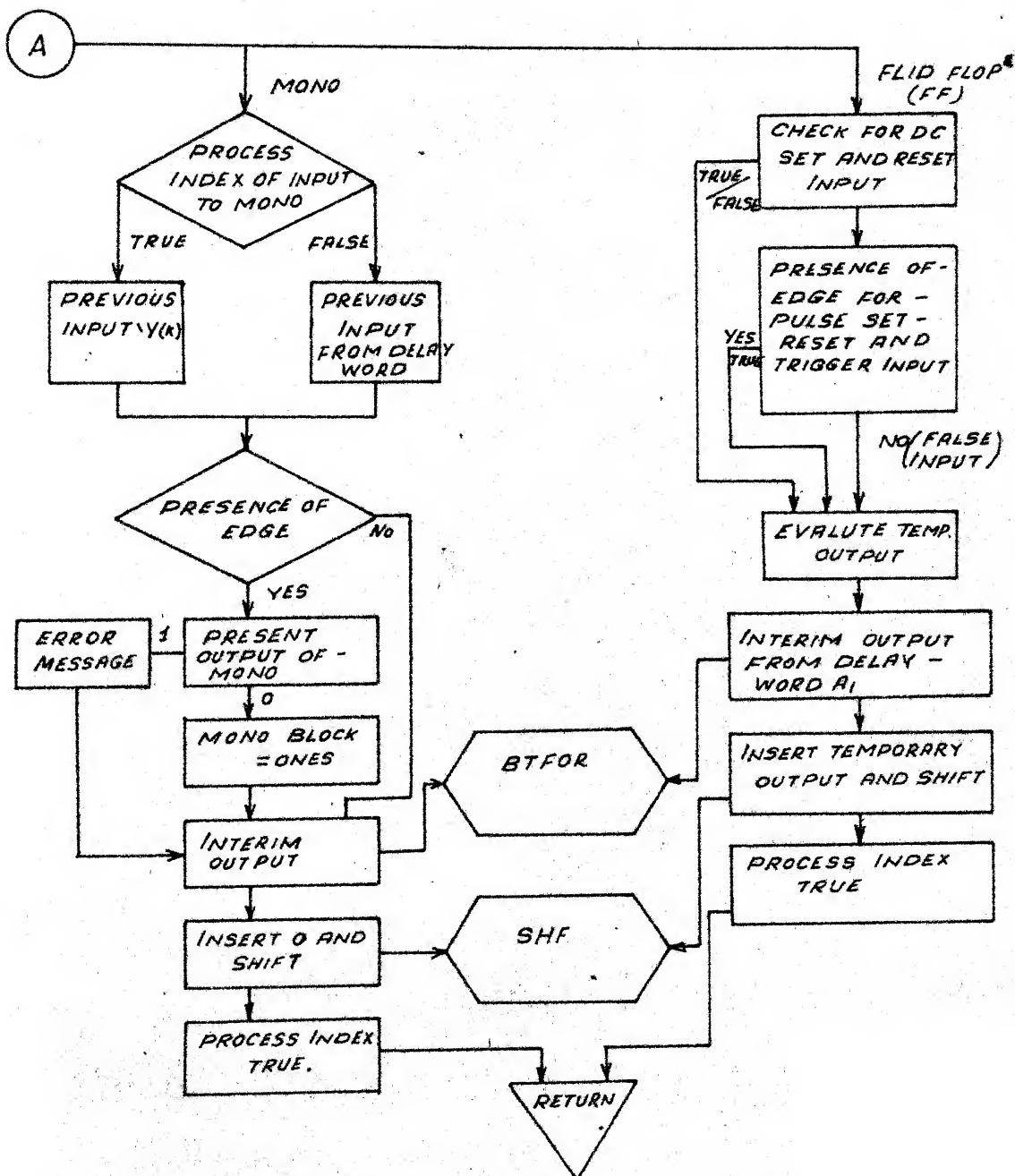


FIG. 5-1 (b) (CONTINUED)

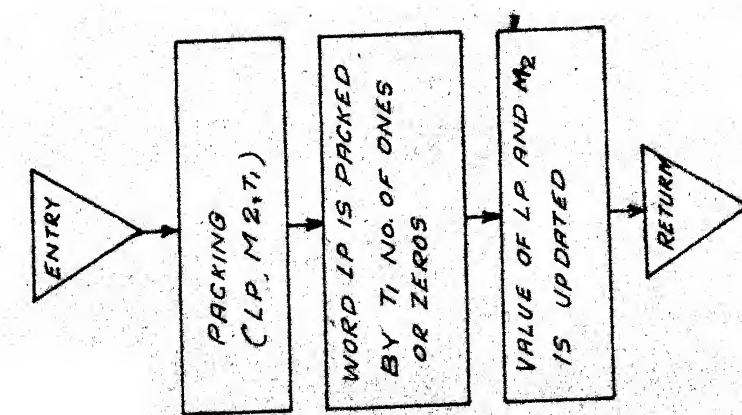


FIG. 5.1(C) PACKING ROUTINE

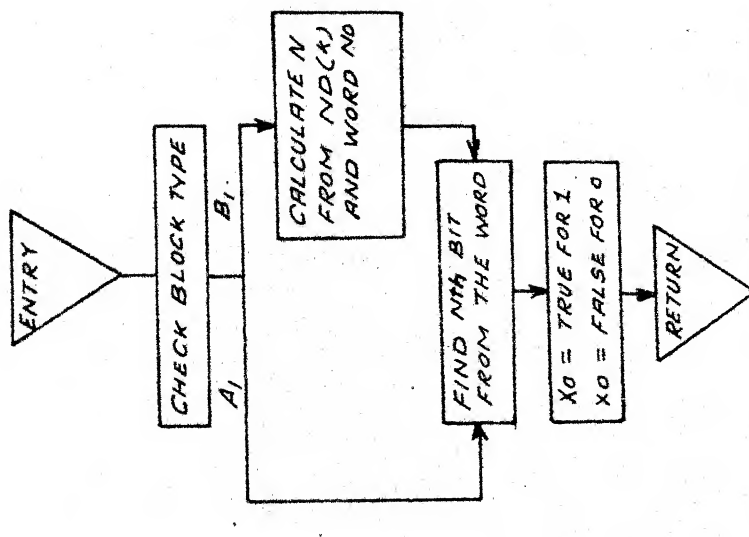


FIG 5(d) BTFFDR ROUTINE

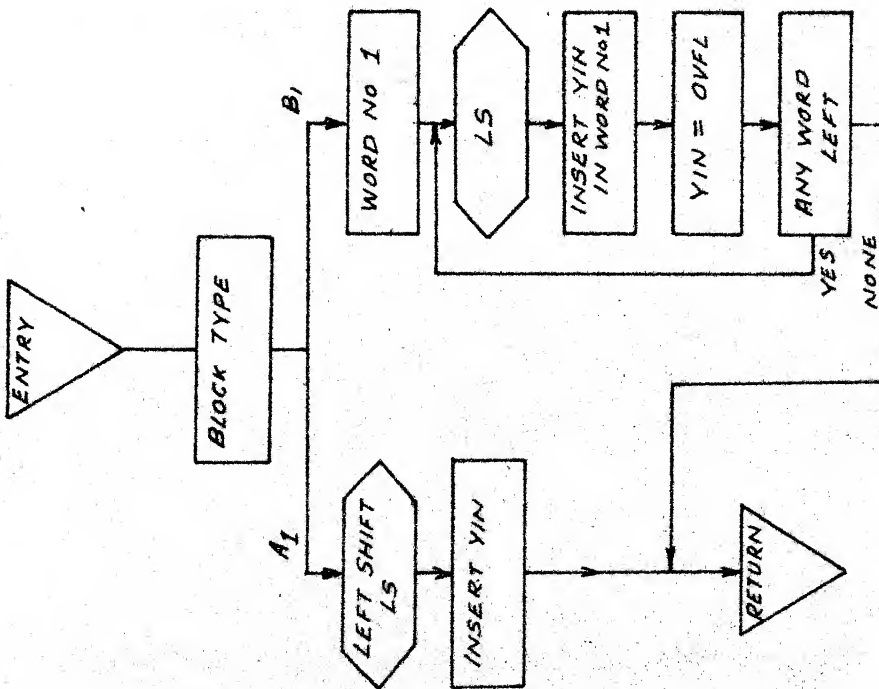


FIG. 5-1(d) SHF ROUTINE

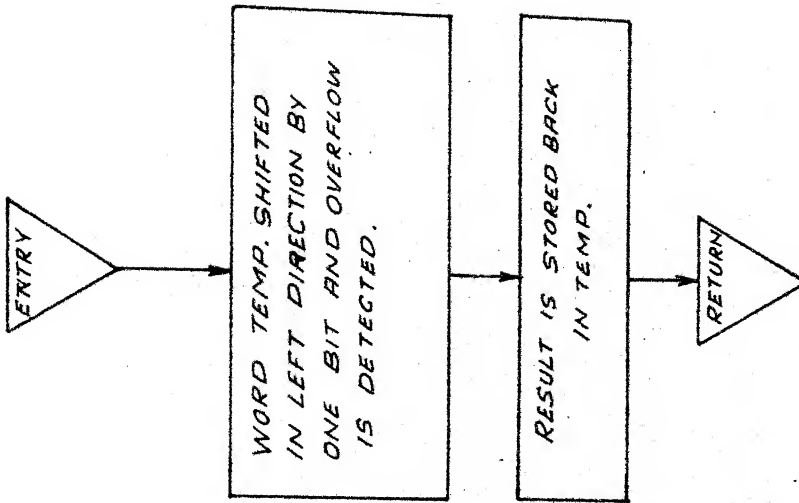


FIG. 5-1(e) LS ROUTINE

## \*\* DIGITAL LOGIC SIMULATION \*\*

LIBFTC MAIN2

## MAIN PROGRAM

```

*****
DIMENSION NI(100,5),ND(100),NT(100),A(100),B(50,10),Z(100),Y(100)
DIMENSION BIT(36),FMT(12),FMT1(12),FMT2(12),DIGIT(10),ARRAY(36)
DIMENSION NO(25),C(360),NIP(25),TO(25),TI(25),PROC(100)
COMMON NI,NT,ND,A,B,TI,ARRAY,BIT,Z,Y,DIGIT,INPWD,PROC,EDGE
INTEGER A,BIT,YIN,BN,DIGIT,ARRAY,PIN,TI,TO,T1,M2,C,OP1,OP2,OP3,OP4
INTEGER CR(10)
INTEGER BLANK
REAL INPUT
COMMON/SW/SWITCH
LOGICAL Z,Y,XO,ZT, LOGIC,FBSEQ,START,RESTRT,PROC,SWITCH
LOGICAL EDGE

```

```

-----
1 FORMAT(1X,012)
2 FORMAT(1X,11)
3 FORMAT(7(I3,4X))
4 FORMAT(20X,7(I3,4X))
5 FORMAT(12A6)
6 FORMAT(A6)
7 FORMAT(20X,A6)
8 FORMAT(I3,71X,A6)
9 FORMAT(20X,I3,71X,A6)
10 FORMAT(I3,2X,A6,2X,6(I3,1X),37X,A6)
11 FORMAT(20X,I3,2X,A6,2X,6(I3,1X),37X,A6)
12 FORMAT(I3,2X,A6,2X,I3,2X,L6,50X,A6)
13 FORMAT(80A1)
14 FORMAT(1X,10(012,1X))
15 FORMAT(5(L6,6X))
16 FORMAT(1X,I3)
17 FORMAT(17X,5(L6,2X))
18 FORMAT (20X,I3,2X,A6,2X,I3,2X,L6,50X,A6)
19 FORMAT(20X,80A1)
20 FORMAT (20X,12A6)
21 FORMAT(20X,*LOGIC      START      RESTRT      EDGE      SWITCH*/20X,*-----
1-----* )
22 FORMAT(20X,*KIMAX      NOMAX      KMAX      INVM      INPWD      INPTM      KIDMX */20X,
1*-----* )
23 FORMAT(20X,*CIRCUIT CONFIGURATION*)
24 FORMAT(20X,*OUTPUT FORMAT *)
25 FORMAT(20X,*INPUT WAVEFORM*)
26 FORMAT(20X,5(012,1X)/20X,5(012,1X))
27 FORMAT(20X,*WAVEFORMS OF INPUT AND OUTPUT BLOCKS*)
28 FORMAT(1H1)

```



## \*\* DIGITAL LOGIC SIMULATION \*\*

```

1006 READ(5) (Z(K),K=2,KMAX)
      READ(5) (A(K),K=KN,KMAX)
      N=1
1007 READ(5) (B(N,J),J=1,INPWD)
      N=N+1
      IF(N.GT.KIDMX) GOTO 1004
      GOTO 1007

```

```

C-----
C      INITIALIZATION OF CONSTANTS
C      BIT(36) ALL DIAGONAL ELEMENTS ARE 1.
C-----

```

```

1004 BIT(1)=1
      IF(SWITCH) PRINT1,BIT(1)
      DO 1005 I=2,35
      BIT(I)=2** (I-1)
      IF(SWITCH) PRINT1,BIT(I)
1005 CONTINUE
      BIT(36)=-0
      IF(SWITCH) PRINT1,BIT(36)
C      DIGIT(10)      DECIMAL ARRAY
      DO 1010 I=1,10
      DIGIT(I)=I-1
      IF(SWITCH) PRINT 2,DIGIT(I)

```

```

1010 CONTINUE
C      ARRAY(36)      TRIANGULAR ARRAY
      ARRAY(1)=1
      IF(SWITCH) PRINT 1,ARRAY(1)
      DO 1015 I=2,35
      ARRAY(I)=2** (I-1)+ARRAY(I-1)
      IF(SWITCH) PRINT 1,ARRAY(I)
1015 CONTINUE
      ARRAY(36)=10NES
      IF(SWITCH) PRINT1,ARRAY(36)

```

```

C-----
C      READ AND PRINT DATA
      PRINT 29
      PRINT 24
      READ 5,FMT
      PRINT 20,FMT
      READ 5,FMT1
      PRINT20,FMT1
      READ5,FMT2
      PRINT 20,FMT2
      PRINT 29
      PRINT23

```

```

1020 READ6,HEAD
      PRINT 7,HEAD
      IF(HEAD.EQ.OUTPUT) GOTO 1025
      IF(HEAD.EQ.CIRCUIT) GOTO 1030
      IF(HEAD.EQ.INPUT) GOTO 1040

```

## \*\* DIGITAL LOGIC SIMULATION \*\*

```

1025  KO=1
1026  READ 8,NO(KO),BCNTRL
      PRINT9,NO(KO),BCNTRL
      KO=KO+1
      IF(BCNTRL.NE.ZERO) GOTO 1026
      GOTO 1020
1030  READ 10,K,NT(K),ND(K),(NI(K,J),J=1,5),CCNTRL
      PRINT 11,K,NT(K),ND(K),(NI(K,J),J=1,5),CCNTRL
      IF(CCNTRL.NE.CEND) GOTO 1030
      GO TO 1020
C-----
C
C  INPUT
C  INPUT WAVEFORM CONFIGURATION IS READ AND PACKED IN BLOCK TYPE B(K)
C-----
1040  READ 12,K,NT(K),ND(K),FBSEQ,CNTRL
      PRINT 18,K,NT(K),ND(K),FBSEQ,CNTRL
      READ 13,(C(I),I=1,INVM)
      PRINT 19,(C(I),I=1,INVM)
C  FBSEQ  T----FIRST INPUT IS AT 1 LEVEL
C          F----FIRST INPUT IS AT 0 LEVEL
      M2=36
      LP=1
      ID=1
1045  T1=0
      KD=1
      IF(C(ID).EQ.BLANK)GO TO 1060
1050  IF(C(ID).EQ.CR(KD)) GOTO 1055
      KD=KD+1
      IF(KD.EQ.10)GO TO 1055
      GO TO 1050
1055  T1=DIGIT(KD)*100
      IF(SWITCH) PRINT 16,T1
1060  ID=ID+1
      KD=1
      IF(C(ID).EQ.BLANK)GO TO 1075
1065  IF(C(ID).EQ.CR(KD)) GOTO 1070
      KD=KD+1
      IF(KD.EQ.10)GO TO 1070
      GO TO 1065
1070  T1=T1+DIGIT(KD)*10
      IF(SWITCH) PRINT 16,T1
1075  ID=ID+1
      KD=1
      IF(C(ID).EQ.BLANK) GOTO 1096
1080  IF(C(ID).EQ.CR(KD)) GOTO 1085
      KD=KD+1
      IF(KD.EQ.10)GO TO 1085
      GO TO 1080
1085  T1=T1+DIGIT(KD)

```

## \*\* DIGITAL LOGIC SIMULATION \*\*

```

IF(SWITCH) PRINT 16,T1
IF(BCNTRL.NE.ZERO) GOTO 1020
IF(FBSEQ) GOTO 1090
PIN=0
GOTO 1095
1090 PIN=1
1095 CALLPAKING(K,LP,PIN,M2)
ID=ID+1
IF(SWITCH) PRINT14,(B(K,J),J=1,10)
IF(ID.GT.INVM) GOTO1096
FBSEQ=.NOT. FBSEQ
GOTO 1045
1096 PRINT 29
PRINT 25
PRINT 26,(B(K,J),J=1,10)
IF(CNTRL.NE.END) GOTO 1040
C-----
C EVALUATION STARTS
C-----
DO 3005 L=2,KIMAX
NIP(L)=L
3005 CONTINUE
1100 INPT=1
1150 PRINT 28
IPRNT=1
PRINT 27
PRINT FMT1
PRINT FMT2,(NIP(L),L=2,KIMAX),(NO(L),L=1,NOMAX)
1105 DO1110 K=2,KMAX
IF(NT(K).EQ.IINP) CALL INP(K)
IF(NT(K).EQ.IAND)CALL ANDD(K)
IF(NT(K).EQ.INOT) CALLNOT(K)
IF(NT(K).EQ.IOR) CALLORR(K)
IF(NT(K).EQ.IFF)CALL FF(K)
IF(NT(K).EQ.INOR) CALL NOR(K)
IF(NT(K).EQ.IMONO) CALL MONO(K)
IF(NT(K).EQ.IDELAY) CALL DELAY(K)
1110 CONTINUE
C-----
C
C OUTPUT ROUTINE
C-----
DO3000 I=2,KMAX
3000 Z(I)=Y(I)
DO 3010 K=1,NOMAX
NEW=NO(K)
IF(LOGIC) GOTO 3030
IF(Z(NEW)) GOTO 3020
TO(K)= OP3

```

## \*\* DIGITAL LOGIC SIMULATION \*\*

```

      GOTO 3010
3020  TO(K)=OP4
      GOTO 3010
3030  IF(Z(NEW)) GOTO 3040
      TO(K)=OP1
      GOTO 3010
3040  TO(K)=OP2
3010  CONTINUE
      DO 3060 I=2,KIMAX
      IF(LOGIC) GOTO 3070
      IF(Z(I)) GOTO 3080
      TI(I)=OP3
      GOTO 3060
3080  TI(I)=OP4
      GOTO 3060
3070  IF(Z(I)) GOTO 3090
      TI(I)=OP1
      GOTO 3060
3090  TI(I)=OP2
3060  CONTINUE
      PRINTFMT,INPT,(TI(I),I=2,KIMAX),(TO(K),K=1,NOMAX)
      INPT=INPT+1
      IPRNT=IPRNT+1
      IF(IPRNT.GT.43) GOTO 1150
3076  IF(INPT.GT.INPTM) GOTO 3095
      DO 3050 K=2,KMAX
3050  PROC(K)=.FALSE.
      GOTO 1105
3095  IF(.NOT.RESTRT)GOTO 3990
C
      WRITE(7) (Z(K),K=2,KMAX)
      WRITE(7) (A(K),K=KN,KMAX)
3096  NOUT=I
      WRITE(7) (B(NOUT,J),J=1,INPWD)
      NOUT=NOUT+1
      IF(NOUT.GT.KIDMX) GOTO 3990
      GOTO 3096
3990  STOP
      END

```

## \*\* DIGITAL LOGIC SIMULATION \*\*

LIBFIC AND2

C\*\*\*\*\*

C

C AND BLOCK

C

C\*\*\*\*\*

SUBROUTINEANDD(K)

DIMENSION NI(100,5),ND(100),NT(100),A(100),B(50,10),Z(100),Y(100)

DIMENSION DIGIT(10),ARRAY(36),BIT(36),PROC(100)

COMMON NI,NT,ND,A,B,T1,ARRAY,BIT,Z,Y,DIGIT,INPWD,PROC,EDGE

COMMON/SW/SWITCH

DATA A1,B1/6HA1,6HB1/

LOGICAL Z,Y,XO,ZT,PROC,SWITCH

LOGICAL EDGE

INTEGER A,BIT,B,BN,DIGIT,ARRAY,YIN

101 FORMAT(1X,L1)

NK=ND(K)

DO150 I=1,5

NR=NI(K,I)

IF(NR.EQ.0) GOTO 135

IF(NR.EQ.1) GOTO 150

IF(NR.EQ.(-1)) GOTO 160

IF(.NOT.Z(NR)) GOTO 135

150 CONTINUE

GOTO 160

135 ZT=.FALSE.

GOTO 165

160 ZT=.TRUE.

165 BN=1

IF(SWITCH) PRINT 101,ZT

BLOCK=A1

IF(SWITCH) PRINT 102,BLOCK

102 FORMAT(1X,D12)

CALL BTFDR(K,BN,XO,BLOCK)

Y(K)=XO

IF(SWITCH) PRINT 101,Y(K)

IF(ZT) GOTO 100

YIN=0

GOTO 110

100 YIN=1

110 CALL SHF(K,BLOCK,YIN)

PROC(K)=.TRUE.

RETURN

END

## \*\* DIGITAL LOGIC SIMULATION \*\*

LIBFTC DELAY2

C\*\*\*\*\*

C

C DELAY BLOCK

C

C\*\*\*\*\*

SUBROUTINE DELAY(K)

DIMENSION NI(100,5),ND(100),NT(100),A(100),B(50,10),Z(100),Y(100)

DIMENSION DIGIT(10),ARRAY(36),BIT(36),PROC(100)

COMMON NI,NT,ND,A,B,T1,ARRAY,BIT,Z,Y,DIGIT,INPWD,PROC,EDGE

COMMON/SW/SWITCH

DATA A1,B1/6HA1,6HB1 /

LOGICAL Z,Y,XO,ZT,PROC,SWITCH

LOGICAL EDGE

INTEGER A,BIT,B,BN,DIGIT,ARRAY,YIN

201 FORMAT(1X,L1)

NK=ND(K)

N1=NI(K,1)

BN=1

BLOCK=B1

IF(SWITCH) PRINT 202,BLOCK

202 FORMAT (1X,012)

CALL BTFR(K,BN,XO,BLOCK)

IF(SWITCH) PRINT 201,Y(K)

Y(K)=XO

IF(Z(N1)) GOTO 210

GOTO 220

210 YIN=1

GOTO 230

220 YIN=0

230 CALL SHE(K,BLOCK,YIN)

PROC(K)=.TRUE.

RETURN

END

## \*\* DIGITAL LOGIC SIMULATION \*\*

LIBFTC DRZ

C\*\*\*\*\*

C

C OR BLOCK

C

C\*\*\*\*\*

SUBROUTINE ORR(K)

DIMENSION NI(100,5),ND(100),NT(100),A(100),B(50,10),Z(100),Y(100)

DIMENSION DIGIT(10),ARRAY(36),BIT(36),PROC(100)

COMMON NI,NT,ND,A,B,T1,ARRAY,BIT,Z,Y,DIGIT,INPWD,PROC,EDGE

COMMON/SW/SWITCH

DATA A1,B1/6HA1,6HB1 /

LOGICAL Z,Y,XD,ZT,PROC,SWITCH

LOGICAL EDGE

INTEGER A,BIT,B,BN,DIGIT,ARRAY,YIN

301 FORMAT(1X,L1)

NK=ND(K)

DO 350 I=1,5

NR=NI(K,I)

IF(NR.EQ.1) GOTO 335

IF(NR.EQ.0) GOTO 350

IF(NR.EQ.(-1)) GOTO 360

IF(Z(NR)) GOTO 335

350 CONTINUE

GOTO 360

335 ZT=.TRUE.

GOTO 365

360 ZT=.FALSE.

365 BN=1

IF(SWITCH) PRINT 301,ZT

BLOCK=A1

IF(SWITCH) PRINT 302,BLOCK

302 FORMAT (1X,D12)

CALL BTFDR (K,BN,XD,BLOCK)

IF(SWITCH) PRINT 301,Y(K)

Y(K)=XD

IF(ZT) GOTO 300

YIN=0

GOTO 310

300 YIN=1

310 CALL SHF(K,BLOCK,YIN)

PROC(K)=.TRUE.

RETURN

END

\*\* DIGITAL LOGIC SIMULATION \*\*

LIBFTC FF2

C\*\*\*\*\*

C

C FLIP FLOP BLOCK

C

C\*\*\*\*\*

SUBROUTINE FF(K)

DIMENSION NI(100,5),ND(100),NT(100),A(100),B(50,10),Z(100),Y(100)

DIMENSION DIGIT(10),ARAY(36),BIT(36),PROC(100)

COMMON NI,NT,ND,A,B,T1,ARAY,BIT,Z,Y,DIGIT,INPND,PROC,EDGE

COMMON/SW/SWITCH

DATA A1,B1/6HA1 ,6HB1 /

DATA IMONO,IDELAY,IINP/6HMONO ,6HDELAY ,6HINP

LOGICAL Z,Y,XO,ZT,PROC,SWITCH

LOGICAL EDGE

LOGICAL ZN1,ZN2,ZN3,ZN4,ZN5

INTEGER A,BIT,B,BN,DIGIT,ARAY,YIN

481 FORMAT(1X,5L1)

482 FORMAT(1X,L1)

C N1 DC SET

C N2 DC RESET

C N3 PULSE SET

C N4 PULSE RESET

C N5 COMPLEMENTARY INPUT

N1=NI(K,1)

IF(N1.EQ.(-1)) GOTO 405

IF(N1.EQ.0) GOTO 405

410 ZN1=.TRUE.

GOTO 415

405 ZN1=.FALSE.

415 N2=NI(K,2)

IF(N2.EQ.(-1)) GOTO 425

IF(N2.EQ.0) GOTO 425

420 ZN2=.TRUE.

GOTO 430

425 ZN2=.FALSE.

430 N3=NI(K,3)

IF(N3.EQ.(-1)) GOTO 440

BN=1

IF((NT(N3).EQ.IINP).OR.(NT(N3).EQ.IDELAY).OR.(NT(N3).EQ.IMONO))

1GOTO 431

GOTO 432

431 BLOCK=B1

GOTO 433

432 BLOCK=A1

433 IF(PROC(N3)) GOTO 436

CALL BTFR(N3,BN,XO,BLOCK)

GOTO 437

436 XO=Y(N3)

437 IF(EDGE) GOTO 438

## \*\* DIGITAL LOGIC SIMULATION \*\*

```

      IF(Z(N3).AND..NOT.XO) GOTO 435
      GOTO 440
438 IF(.NOT.Z(N3).AND.XO) GOTO 435
      GOTO 440
435 ZN3=.TRUE.
      GOTO 445
440 ZN3=.FALSE.
445 N4=NI(K,4)
      IF(N4.EQ.(-1)) GOTO 455
      BN=1
      IF((NT(N4).EQ.IINP).OR.(NT(N4).EQ.IDELAY).OR.(NT(N4).EQ.IMONO))
1GOTO 446
      GOTO 447
446 BLOCK=B1
      GOTO 448
447 BLOCK=A1
448 IF(PROC(N4)) GOTO 451
      CALL BTFDR(N4,BN,XO,BLOCK)
      GOTO 452
451 XO=Y(N4)
452 IF(EDGE) GOTO 453
      IF(Z(N4).AND..NOT.XO) GOTO 450
      GOTO 455
453 IF(.NOT.Z(N4).AND.XO) GOTO 450
      GOTO 455
450 ZN4=.TRUE.
      GOTO 460
455 ZN4=.FALSE.
460 N5=NI(K,5)
      IF(N5.EQ.(-1)) GOTO 465
      BN=1
      IF((NT(N5).EQ.IINP).OR.(NT(N5).EQ.IDELAY).OR.(NT(N5).EQ.IMONO))
1GOTO 461
      GOTO 462
461 BLOCK=B1
      GOTO 463
462 BLOCK=A1
463 IF(PROC(N5)) GOTO 466
      CALL BTFDR (N5,BN,XO,BLOCK)
      GOTO 467
466 XO=Y(N5)
467 IF(EDGE) GOTO 468
      IF(Z(N5).AND..NOT.XO) GOTO 465
      GOTO 470
468 IF(.NOT.Z(N5).AND.XO) GOTO 465
      GOTO 470
465 ZN5=.TRUE.
      GOTO 475
470 ZN5=.FALSE.
475 BN=0

```

## \*\* DIGITAL LOGIC SIMULATION \*\*

```
IF(SWITCH) PRINT 481,ZN1,ZN2,ZN3,ZN4,ZN5
BLOCK=A1
CALL BTFDR(K,BN,XO,BLOCK)
IF((ZN1.OR.ZN3).OR.(.NOT.XO.AND.ZN5).OR.(XO.AND..NOT.ZN5.AND..
1(ZN2.OR.ZN4))) GOTO 480
ZT=.FALSE.
GOTO 485
480 ZT=.TRUE.
485 BN=1
IF(SWITCH) PRINT 482,ZT
BLOCK=A1
CALL BTFDR(K,BN,XO,BLOCK)
Y(K)=XO
IF(SWITCH) PRINT 482,Y(K)
IF(ZT) GOTO 400
YIN=0
GOTO 401
400 YIN=1
401 CALL SHF (K,BLOCK,YIN)
PROC(K)=.TRUE.
RETURN
END
```

## \*\* DIGITAL LOGIC SIMULATION \*\*

LIBFTC NOR2

C\*\*\*\*\*

C

C NOR BLOCK

C

C\*\*\*\*\*

SUBROUTINE NOR(K)

DIMENSION NI(100,5),ND(100),NT(100),A(100),B(50,10),Z(100),Y(100)

DIMENSION DIGIT(10),ARRAY(36),BIT(36),PROC(100)

COMMON NI,NT,ND,A,B,T1,ARRAY,BIT,Z,Y,DIGIT,INPWD,PROC,EDGE

COMMON/SW/SWITCH

DATA A1,B1/6HA1 ,6HB1 /

LOGICAL Z,Y,XO,ZT,PROC,SWITCH

LOGICAL EDGE

INTEGER A,BIT,B,BN,DIGIT,ARRAY,YIN

501 FORMAT(1X,L1)

NK=ND(K)

DO 550 I=1,5

NR=NI(K,I)

IF(NR.EQ.0) GOTO 550

IF(NR.EQ.1) GOTO 535

IF(NR.EQ.(-1)) GOTO 540

IF(Z(NR)) GOTO 535

550 CONTINUE

540 ZT=.TRUE.

GOTO 560

535 ZT=.FALSE.

560 BN=1

IF(SWITCH) PRINT 501,ZT

BLOCK=A1

IF(SWITCH) PRINT 502,BLOCK

502 FORMAT (1X,D12)

CALL BTFDR(K,BN,XO,BLOCK)

Y(K)=XO

IF(SWITCH) PRINT 501,Y(K)

IF(ZT) GOTO 500

YIN=0

GOTO 510

500 YIN=1

510 CALL SHE(K,BLOCK,YIN)

PROC(K)=.TRUE.

RETURN

END

## \*\* DIGITAL LOGIC SIMULATION \*\*

LIBFTC MONO2

```

C *****
C
C   MONOSTABLE BLOCK
C *****
C   SUBROUTINE MONO(K)
C   DIMENSION NI(100,5),ND(100),NT(100),A(100),B(50,10),Z(100),Y(100)
C   DIMENSION DIGIT(10),ARRAY(36),BIT(36),PROC(100)
C   COMMON NI,NT,ND,A,B,T1,ARRAY,BIT,Z,Y,DIGIT,INPWD,PROC,EDGE
C   COMMON/SW/SWITCH
C   LOGICAL Z,Y,XO,ZT,PROC,SWITCH
C   LOGICAL EDGE
C   DATA A1,B1/6HA1      ,6HB1      /
C   DATA IMONO,IDELAY,IINP/6HMONO    ,6HDELAY ,6HINP    /
C   DATA IONES/0777777777777777/
C   INTEGER A,BIT,B,BN,DIGIT,ARRAY,YIN
601  FORMAT(1X,L1)
      NK=ND(K)
      N1=NI(K,1)
      BN=1
      IF((NT(N1).EQ.IINP).OR.(NT(N1).EQ.IDELAY).OR.(NT(N1).EQ.IMONO))
1GOTO 650
      BLOCK=A1
      GOTO 655
650  BLOCK=B1
655  IF(PROC(N1)) GOTO 630
      CALL BTFR(K,BN,XO,BLOCK)
      GOTO 635
630  XO=Y(N1)
635  IF(EDGE) GOTO 636
      IF(Z(N1).AND..NOT.XO) GOTO 615
      GOTO 610
636  IF(.NOT.Z(N1).AND.XO) GOTO 615
      GOTO 610
615  BN=1
      BLOCK=B1
      CALL BTFR(K,BN,XO,BLOCK)
      IF(XO) GOTO 680
      DO 605 I=1,INPWD
605  B(K,I)=IONES
      GOTO 610
680  PRINT 602,K
602  FORMAT(20X,*WARNING MESSAGE*/ 20X,*PULSE REPITION TIME LESS THAN
1 MONOSTABLE PULSE DURATION*/20X,*IGNORED*,*BLOCK NO.  *,13)
610  BN=1
      BLOCK=B1
      CALL BTFR(K,BN,XO,BLOCK)
      Y(K)=XO
      IF(SWITCH) PRINT 601,Y(K)

```

\*\* DIGITAL LOGIC SIMULATION \*\*

```
YIN=0  
CALL SHF(K,BLOCK,YIN)  
PROC(K)=.TRUE.  
RETURN  
END
```

## \*\* DIGITAL LOGIC SIMULATION \*\*

LIBFTC NOT2

```

C *****
C
C     NOT BLOCK
C
C *****
  SUBROUTINE NOT(K)
    DIMENSION NI(100,5),ND(100),NT(100),A(100),B(50,10),Z(100),Y(100)
    DIMENSION DIGIT(10),ARAY(36),BIT(36),PROC(100)
    COMMON/SW/SWITCH
    COMMON NI,NT,ND,A,B,T1,ARAY,BIT,Z,Y,DIGIT,INPWD,PROC,EDGE
    INTEGER A,BIT,B,BN,DIGIT,ARAY,YIN
    DATA A1,B1/6HA1,6HB1 /
    LOGICAL Z,Y,XO,ZT,PROC,SWITCH
    LOGICAL EDGE
701  FORMAT(1X,L1)
    NI=NI(K,1)
    NK=ND(K)
    IF(Z(NI)) GOTO 770
    GOTO 775
770  ZT=.FALSE.
    GOTO 780
775  ZT=.TRUE.
780  BN=1
    IF(SWITCH) PRINT 701,ZT
    BLOCK=A1
    IF(SWITCH) PRINT 702 , BLOCK
702  FORMAT(1X,O12)
    CALL BTFDR(K,BN,XO,BLOCK)
    Y(K)=XO
    IF(SWITCH) PRINT 701,Y(K)
    IF(ZT) GOTO 710
    YIN=0
    GOTO 715
710  YIN=1
715  CALL SHF (K,BLOCK,YIN)
    PROC(K)=.TRUE.
    RETURN
  END

```

## \*\* DIGITAL LOGIC SIMULATION \*\*

SIBFTC BTFDR2

C\*\*\*\*\*

C

C BTFDR ROUTINE      A1 ONE WORD DELAY    B1 MULTIWORD DELAY  
C K=BLOCK NO.    BN    0 FIRST BIT    1    NTH BIT  
C

C\*\*\*\*\*

SUBROUTINE BTFDR(K,BN,XO,BLOCK)

DIMENSION NI(100,5),ND(100),NT(100),A(100),B(50,10),Z(100),Y(100)

DIMENSION DIGIT(10),ARAY(36),BIT(36),PROC(100)

COMMON NI,NT,ND,A,B,T1,ARAY,BIT,Z,Y,DIGIT,INPWD,PROC,EDGE

INTEGER A,BIT,B,BN,DIGIT,ARAY,YIN

COMMON/SW/SWITCH

DATA A1,B1/6HA1      ,6HB1      /

LOGICAL Z,Y,XO,ZT,PROC,SWITCH

LOGICAL EDGE

801 FORMAT(1X,L1)

IF(BLOCK.EQ.A1)GO TO 800

IF(BLOCK.EQ.B1)GO TO 870

800 NK=ND(K)

IF(BN.EQ.0)GO TO 850

AR1=AND(BIT(NK),A(K))

IF(AR1.EQ.0.) GOTO 810

XO=.TRUE.

GO TO 890

810 XO=.FALSE.

GO TO 890

850 ARO=AND(BIT(1),A(K))

IF(ARO.EQ.0.)GO TO 860

XO=.TRUE.

GO TO 890

860 XO=.FALSE.

GO TO 890

870 NL=ND(K)

IF(BN.EQ.0)GO TO 880

NNE=NL/36

NL1=NL-(NL/36)\*36

NRL=INPWD-NNE

BR1=AND(BIT(NL1),B(K,NRL))

IF(BR1.EQ.0.)GO TO 820

XO=.TRUE.

GO TO 890

820 XO=.FALSE.

GO TO 890

880 BRO=AND(BIT(1),B(K,INPWD))

IF(BRO.EQ.0.)GO TO 885

XO=.TRUE.

GO TO 890

885 XO=.FALSE.

890 IF(SWITCH) PRINT 801,XO

. \*\* DIGITAL LOGIC SIMULATION \*\*

RETURN  
END

## \*\* DIGITAL LOGIC SIMULATION \*\*

LIBFIC INP2

C\*\*\*\*\*

C

C

C

C

C

C\*\*\*\*\*

INPUT GENERATOR

THIS ROUTINE FINDS INPUT BIT FROM THE INPUT BLOCK,

INSERTS DAT THE FIRST BIT AND THEN SHIFTS WHOLE BLOCK BY ONE BIT

SUBROUTINE INP(K)

DIMENSION NI(100,5),ND(100),NT(100),A(100),B(50,10),Z(100),Y(100)

DIMENSION DIGIT(10),ARRAY(36),BIT(36),PROC(100)

COMMON NI,NT,ND,A,B,T1,ARRAY,BIT,Z,Y,DIGIT,INPWD,PROC,EDGE

INTEGER A,BIT,B,BN,DIGIT,ARRAY,YIN

COMMON/SW/SWITCH

LOGICAL Z,Y,XO,ZT,PROC,SWITCH

LOGICAL EDGE

DATA A1,B1/6HA1 ,6HB1 /

901 FORMAT(1X,L1)

INSN=ISIGN(1,B(K,1))

IF(INSN.EQ.(-1)) GOTO 910

Y(K)=.FALSE.

GOTO 990

910 Y(K)=.TRUE.

GOTO 990

990 IF(SWITCH) PRINT 901,Y(K)

YIN=0

BLOCK=B1

IF(SWITCH) PRINT 902, BLOCK

902 FORMAT(1X,O12)

CALL SHF(K,BLOCK,YIN)

PROC(K)=.TRUE.

RETURN

END

## \*\* DIGITAL LOGIC SIMULATION \*\*

LIBFTC LS2

```

C*****
C
C      LEFT SHIFT ROUTINE
C      THIS SUBROUTINE SHIFTS CONTENTS OF TEMP BY ONE BIT, DETECTS O
C      UR 1 AS OVERFLOW AND PLACES BACK INTO TEMP
C      TEMP      ONE WORD
C*****
C      SUBROUTINE LS(TEMP,OVFL)
C      DIMENSION NI(100,5),ND(100),NT(100),A(100),B(50,10),Z(100),Y(100)
C      DIMENSION DIGIT(10),ARRAY(36),BIT(36),PROC(100)
C      COMMON NI,NT,ND,A,B,TI,ARRAY,BIT,Z,Y,DIGIT,INPWD,PROC,EDGE
C      DATA IO4/04000000000000/
C      INTEGER A,BIT,B,BN,DIGIT,ARRAY,YIN
C      INTEGER TEMP,OVFL,C2P34
C      DATA C2P34/02000000000000/
C      COMMON/SW/SWITCH
C      LOGICAL Z,Y,XO,ZT,PROC,SWITCH
C      LOGICAL EDGE
4001  FORMAT(1X,012,2X,11)
4002  FORMAT(1X,012)
      IF(SWITCH) PRINT 4002,TEMP
      OVFL=0
      NOVFL= ISIGN(1,TEMP)
      IF(NOVFL.EQ.(-1)) OVFL=1
      TEMP=IABS(TEMP)
      IF(TEMP.EQ.0)GO TO4090
      IF(TEMP.EQ.C2P34) GOTO 4010
      MS=0
      IF(TEMP.GE.C2P34)MS=1
      TEMP=(TEMP-MS*C2P34)*2
      IF(MS.EQ.1)TEMP=-TEMP
      GOTO 4020
4010  TEMP=IO4
4020  IF(SWITCH) PRINT 4001,TEMP,OVFL
4090  RETURN
      END

```

## \*\* DIGITAL LOGIC SIMULATION \*\*

LIBFTC SHF2

C\*\*\*\*\*

C  
C SHIFT ROUTINE  
C

C\*\*\*\*\*

```

SUBROUTINE SHF(K,BLOCK,YIN)
  DIMENSION NI(100,5),ND(100),NT(100),A(100),B(50,10),Z(100),Y(100)
  COMMON NI,NT,ND,A,B,T1,ARRAY,BIT,Z,Y,DIGIT,INPWD,PROC,EDGE
  DIMENSION DIGIT(10),ARRAY(36),BIT(36),PROC(100)
  DATA A1,B1/6HA1      ,6HB1      /
  COMMON/SW/SWITCH
  INTEGER A,BIT,B,BN,DIGIT,ARRAY,YIN,TEMP,OVFL
  LOGICAL Z,Y,XO,ZT,PROC,SWITCH
  LOGICAL EDGE
5001 FORMAT(1X,D12)
5002 FORMAT(1X,10012)
  IF(SWITCH) PRINT 5001,BLOCK
  IF(BLOCK.EQ.B1) GOTO 5050
5000 TEMP=A(K)
  IF(SWITCH) PRINT 5001,A(K)
  CALL LS(TEMP,OVFL)
  MS=1
  MMS=ISIGN(1,TEMP)
  IF(MMS.EQ.(-1)) MS=-1
  A(K)=(IABS(TEMP)+YIN)*MS
  IF(SWITCH) PRINT 5001,A(K)
  GOTO 5095
5050 J=INPWD
5060 TEMP=B(K,J)
  IF(SWITCH) PRINT 5001,TEMP
  CALL LS(TEMP,OVFL)
  MS=1
  MMS=ISIGN(1,TEMP)
  IF(MMS.EQ.(-1)) MS=-1
  B(K,J)=(IABS(TEMP)+YIN)*MS
  YIN=OVFL
  J=J-1
  IF(J.EQ.0) GOTO 5090
  GOTO 5060
5090 IF(SWITCH) PRINT 5002,(B(K,J),J=1,INPWD)
5095 RETURN
END

```

## \*\* DIGITAL LOGIC SIMULATION \*\*

LIBFTC PAKNG2

C\*\*\*\*\*

C THIS ROUTINE PACKS THE INPUT STRING  
 C K= BLOCK NO. LP= WORD NO. WHERE PACKING HAS TO BE DONE  
 C PIN= 1 /R 0 M2=NO. OF BITS LEFT IN THE WORD NO. LP  
 C

C\*\*\*\*\*

```

SUBROUTINE PAKING(K,LP,PIN,M2)
  DIMENSION NI(100,5),ND(100),NT(100),A(100),B(50,10),Z(100),Y(100)
  DIMENSION DIGIT(10),ARRAY(36),BIT(36),PROC(100)
  COMMON NI,NT,ND,A,B,T1,ARRAY,BIT,Z,Y,DIGIT,INPWD,PROC,EDGE
  DATA LONES/0777777777777777/
  COMMON/SW/SWITCH
  LOGICAL Z,Y,XO,ZT,PROC,SWITCH
  LOGICAL EDGE
  INTEGER A,BIT,B,BN,DIGIT,ARRAY,YIN,PIN,TNEW,T1,M2
7001 FORMAT(1X,2(I3,1X),0I2)
  NS=ISIGN(1,B(K,LP))
7002 FORMAT(1X,I2)
  IF(SWITCH) PRINT 7002,NS
  IF(T1.LT.M2) GOTO 7000
  IF(T1.EQ.M2) GOTO 7010
  IF(T1.GT.M2) GOTO 7020
7000 IF(PIN.EQ.0) GOTO 7005
  MTR1=M2-T1
  IF(M2.EQ.36) GOTO 7006
  B(K,LP)=(IABS(B(K,LP))+ARRAY(M2)-ARRAY(MTR1))
  IF(SWITCH) PRINT 7003,B(K,LP)
7003 FORMAT(1X,0I2)
  B(K,LP)=B(K,LP)*NS
  GOTO 7005
7006 B(K,LP)=-(B(K,LP)+ARRAY(35)-ARRAY(MTR1))
7005 M2=M2-T1
  GOTO 7095
7010 IF(PIN.EQ.0) GOTO 7015
  IF(M2.EQ.36) GOTO 7016
  B(K,LP)=(IABS(B(K,LP))+ARRAY(M2))*NS
  GOTO 7015
7016 B(K,LP)=-(B(K,LP)+ARRAY(35))
7015 M2=36
  LP=LP+1
  GOTO 7095
7020 IF(PIN.EQ.0) GOTO 7025
  IF(M2.EQ.36) GOTO 7026
  B(K,LP)=(IABS(B(K,LP))+ARRAY(M2))*NS
  GOTO 7025
7026 B(K,LP)=-(B(K,LP)+ARRAY(35))
7025 LP=LP+1
  TNEW=T1-M2

```

## \*\* DIGITAL LOGIC SIMULATION \*\*

```
LIN=TNEW/36
IF(LIN.EQ.0) GOTO 7030
GOTO 7050
7030 IF(PIN.EQ.0) GOTO 7035
MTR2=36-TNEW
B(K,LP)=- (B(K,LP)+ARRAY(35)-ARRAY(MTR2))
7035 M2=36-TNEW
GOTO 7095
7050 IF(PIN.EQ.0) GOTO 7060
LNEW=LP+LIN-1
DO7080 I=LP,LNEW
B(K,I) =IONES
7080 CONTINUE
GOTO 7090
7060 LP=LP+LIN
MT=TNEW-(TNEW/36)*36
M2=36-MT
GOTO 7095
7090 MT=TNEW-(TNEW/36)*36
MTR3=36-MT
B(K,LP)=- (B(K,LP)+ARRAY(35)-ARRAY(MTR3))
M2=36-MT
LP=LP+LIN
7095 IF(SWITCH) PRINT 7001,M2,LP,B(K,LP)
RETURN
END
```

## APPENDIX 4

### SAMPLE PROBLEMS

1. Binary Adder
2. Binary Counter 4 bit
3. Monostable

## -----D A T A -----

LOGIC	START	RESTR	EDGE	SWITCH
1	F	F	T	F

KIMAX	NOMAX	KMAX	INVM	INPWD	INPTM	KIDMX
4	2	13	27	5	180	4

## OUTPUT FORMAT

(20X,13,3X,3(A4,2X),5X,2(A4,2X))

(20X,43H INPUT

OUTPUT)

(26X,3(13,2X),5X,2(13,4X))

## CIRCUIT CONFIGURATION

## OUTPUT

```

7
13
CIRCUIT
5 AND      1  2  3 -1 -1 -1
6 OR       1  2  3 -1 -1 -1
7 OR       1  5  8 -1 -1 -1
8 AND      1  4  6 -1 -1 -1
9 OR       1  4  6 -1 -1 -1
10 AND     1  5  4 -1 -1 -1
11 NOT     1  7 -1 -1 -1 -1
12 AND     1 11  9 -1 -1 -1
13 OR      1 12 10 -1 -1 -1

```

## INPUT

```

2 INP      180      F
90 90

```

## INPUT WAVEFORM

```

000000000000 000000000000 000000777777 777777777777 777777777777
000000000000 000000000000 000000000000 000000000000 000000000000
3 INP      180      F
40 40 40 40 20

```

## INPUT WAVEFORM

```

000000000000 037777777777 776000000000 000077777777 777774000000
000000000000 000000000000 000000000000 000000000000 000000000000
4 INP      180      F
20 20 20 20 20 20 20 20

```

## INPUT WAVEFORM

```

000000177777 740000007777 776000000377 777700000017 777774000000
000000000000 000000000000 000000000000 000000000000 000000000000

```

END

## WAVEFORMS OF INPUT AND OUTPUT BLOCKS

	INPUT					OUTPUT
	2	3	4	7	13	
1	0	0	0	0	0	
2	0	0	0	0	0	
3	0	0	0	0	0	
4	0	0	0	0	0	
5	0	0	0	0	0	
6	0	0	0	0	0	
7	0	0	0	0	0	
8	0	0	0	0	0	
9	0	0	0	0	0	
10	0	0	0	0	0	
11	0	0	0	0	0	
12	0	0	0	0	0	
13	0	0	0	0	0	
14	0	0	0	0	0	
15	0	0	0	0	0	
16	0	0	0	0	0	
17	0	0	0	0	0	
18	0	0	0	0	0	
19	0	0	0	0	0	
20	0	0	0	0	0	
21	0	0	0	0	0	
22	0	0	1	0	0	
23	0	0	1	0	0	
24	0	0	1	0	0	
25	0	0	1	0	0	
26	0	0	1	0	0	
27	0	0	1	0	0	1
28	0	0	1	0	0	1
29	0	0	1	0	0	1
30	0	0	1	0	0	1
31	0	0	1	0	0	1
32	0	0	1	0	0	1
33	0	0	1	0	0	1
34	0	0	1	0	0	1
35	0	0	1	0	0	1
36	0	0	1	0	0	1
37	0	0	1	0	0	1
38	0	0	1	0	0	1
39	0	0	1	0	0	1
40	0	0	1	0	0	1
41	0	1	0	0	0	1
42	0	1	0	0	0	1
43	0	1	0	0	0	1

## WAVEFORMS OF INPUT AND OUTPUT BLOCKS

	INPUT				OUTPUT	
	2	3	4	7	13	
44	0	1	0	0		1
45	0	1	0	0		1
46	0	1	0	0		1
47	0	1	0	0	0	
48	0	1	0	0	0	
49	0	1	0	0		1
50	0	1	0	0		1
51	0	1	0	0		1
52	0	1	0	0		1
53	0	1	0	0		1
54	0	1	0	0		1
55	0	1	0	0		1
56	0	1	0	0		1
57	0	1	0	0		1
58	0	1	0	0		1
59	0	1	0	0		1
60	0	1	0	0		1
61	0	1	1	0		1
62	0	1	1	0		1
63	0	1	1	0		1
64	0	1	1	0		1
65	0	1	1		1	1
66	0	1	1		1	1
67	0	1	1		1	1
68	0	1	1		1	1
69	0	1	1		1	1
70	0	1	1		1	1
71	0	1	1		1	0
72	0	1	1		1	0
73	0	1	1		1	0
74	0	1	1		1	0
75	0	1	1		1	0
76	0	1	1		1	0
77	0	1	1		1	0
78	0	1	1		1	0
79	0	1	1		1	0
80	0	1	1		1	0
81	0	0	0		1	0
82	0	0	0		1	0
83	0	0	0		1	0
84	0	0	0		1	0
85	0	0	0	0		0
86	0	0	0	0		0

100  
110

# WAVEFORMS OF INPUT AND OUTPUT BLOCKS

	INPUT				OUTPUT	
	2	3	4	7	13	
87	0	0	0	0	0	
88	0	0	0	0	0	
89	0	0	0	0	0	
90	0	0	0	0	0	
91	1	0	0	0	0	
92	1	0	0	0	0	
93	1	0	0	0	0	
94	1	0	0	0	0	
95	1	0	0	0	0	
96	1	0	0	0	0	
97	1	0	0	0	0	
98	1	0	0	0	0	
99	1	0	0	0	0	1
100	1	0	0	0	0	1
101	1	0	1	0	0	1
102	1	0	1	0	0	1
103	1	0	1	0	0	1
104	1	0	1	0	0	1
105	1	0	1		1	1
106	1	0	1		1	1
107	1	0	1		1	1
108	1	0	1		1	1
109	1	0	1		1	1
110	1	0	1		1	1
111	1	0	1		1	0
112	1	0	1		1	0
113	1	0	1		1	0
114	1	0	1		1	0
115	1	0	1		1	0
116	1	0	1		1	0
117	1	0	1		1	0
118	1	0	1		1	0
119	1	0	1		1	0
120	1	0	1		1	0
121	1	1	0		1	0
122	1	1	0		1	0
123	1	1	0		1	0
124	1	1	0		1	0
125	1	1	0		1	0
126	1	1	0		1	0
127	1	1	0		1	0
128	1	1	0		1	0
129	1	1	0		1	0

# WAVEFORMS OF INPUT AND OUTPUT BLOCKS

	INPUT					OUTPUT	
	2	3	4	7	13		
130	1	1	0		1	0	
131	1	1	0		1	0	
132	1	1	0		1	0	
133	1	1	0		1	0	
134	1	1	0		1	0	
135	1	1	0		1	0	
136	1	1	0		1	0	
137	1	1	0		1	0	
138	1	1	0		1	0	
139	1	1	0		1	0	
140	1	1	0		1	0	
141	1	1		1	1	0	
142	1	1		1	1	0	
143	1	1		1	1	0	
144	1	1		1	1	0	
145	1	1	1		1		1
146	1	1	1		1		1
147	1	1	1		1		1
148	1	1	1		1		1
149	1	1	1		1		1
150	1	1	1		1		1
151	1	1	1		1		1
152	1	1	1		1		1
153	1	1	1		1		1
154	1	1	1		1		1
155	1	1	1		1		1
156	1	1	1		1		1
157	1	1	1		1		1
158	1	1	1		1		1
159	1	1	1		1		1
160	1	1	1		1		1
161	1	0	0		1		1
162	1	0	0		1		1
163	1	0	0		1		1
164	1	0	0		1		1
165	1	0	0	0		0	
166	1	0	0	0		0	
167	1	0	0	0		0	
168	1	0	0	0		0	
169	1	0	0	0		0	
170	1	0	0	0		0	
171	1	0	0	0			1
172	1	0	0	0			1

# WAVEFORMS OF INPUT AND OUTPUT BLOCKS

	INPUT					OUTPUT
	2	3	4	7	13	
173	1	0	0	0		1
174	1	0	0	0		1
175	1	0	0	0		1
176	1	0	0	0		1
177	1	0	0	0		1
178	1	0	0	0		1
179	1	0	0	0		1
180	1	0	0	0		1

-----D A T A -----

LOGIC	START	RESTART	EDGE	SWITCH
T	F	F	T	F

KIMAX	NOMAX	KMAX	INVM	INPWD	INPTM	KIDMX
2	4	6	99	1	36	2

OUTPUT FORMAT  
 (20X,13,3X,A4,5X,4(A4,4X))  
 (20X,25H INPUT OUTPUT)  
 (26X,13,5X,4(I3,4X))

CIRCUIT CONFIGURATION  
 OUTPUT

3	FF	1	-1	-1	-1	-1	2
4	FF	1	-1	-1	-1	-1	3
5	FF	1	-1	-1	-1	-1	4
6	FF	1	-1	-1	-1	-1	5

INPUT

2	INP	36	T																
1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	4												

INPUT WAVEFORM

525252525257	000000000000	000000000000	000000000000	000000000000
000000000000	000000000000	000000000000	000000000000	000000000000

## WAVEFORMS OF INPUT AND OUTPUT BLOCKS

INPUT		OUTPUT				
	2	3	4	5	6	
1	1	0	0	0	0	
2	0	1	0	0	0	
3	1	1	1	0	0	
4	0	0	1	1	0	
5	1	0	1	1	1	1
6	0	1	1	1	1	1
7	1	1	0	1	1	1
8	0	0	0	1	1	1
9	1	0	0	1	1	1
10	0	1	0	1	1	1
11	1	1	1	1	1	1
12	0	0	1	0	1	1
13	1	0	1	0	1	1
14	0	1	1	0	1	1
15	1	1	0	0	1	1
16	0	0	0	0	1	1
17	1	0	0	0	1	1
18	0	1	0	0	1	1
19	1	1	1	0	1	1
20	0	0	1	1	1	1
21	1	0	1	1	1	0
22	0	1	1	1	1	0
23	1	1	0	1	1	0
24	0	0	0	1	1	0
25	1	0	0	1	1	0
26	0	1	0	1	1	0
27	1	1	1	1	1	0
28	0	0	1	0	1	0
29	1	0	1	0	1	0
30	0	1	1	0	1	0
31	1	1	0	0	1	0
32	0	0	0	0	1	0
33	1	0	0	0	1	0
34	1	1	0	0	1	0
35	1	1	1	0	1	0
36	1	1	1	1	1	0

-----D A T A -----

LOGIC	START	RESTR	EDGE	SWITCH
T	F	F	T	F

KIMAX	NOMAX	KMAX	INVM	INPWD	INPTM	KIDMX
2	1	3	51	10	360	3

OUTPUT FORMAT

(20X,13,3X,A4,10X,A4)

(20X,25H INPUT OUTPUT)

(26X,13,5X,13,4X)

CIRCUIT CONFIGURATION

OUTPUT

3

CIRCUIT

3 MONO 40 2 -1 -1 -1 -1

INPUT

2 INP 360 F  
1 1 50 1 50 1 50 1 50 1 50 1 50 2

INPUT WAVEFORM

200000000000 000002000000 000000000020 000000000000 000200000000  
000000000200 000000000000 020000000000 000000200000 000000000003

# WAVEFORMS OF INPUT AND OUTPUT BLOCKS

INPUT		OUTPUT	
1	2	3	0
1	0	1	1
2	0	1	1
3	0	1	1
4	0	1	1
5	0	1	1
6	0	1	1
7	0	1	1
8	0	1	1
9	0	1	1
10	0	1	1
11	0	1	1
12	0	1	1
13	0	1	1
14	0	1	1
15	0	1	1
16	0	1	1
17	0	1	1
18	0	1	1
19	0	1	1
20	0	1	1
21	0	1	1
22	0	1	1
23	0	1	1
24	0	1	1
25	0	1	1
26	0	1	1
27	0	1	1
28	0	1	1
29	0	1	1
30	0	1	1
31	0	1	1
32	0	1	1
33	0	1	1
34	0	1	1
35	0	1	1
36	0	1	1
37	0	1	1
38	0	1	1
39	0	1	1
40	0	1	1
41	0	1	1
42	0	0	0
43	0	0	0

# WAVEFORMS OF INPUT AND OUTPUT BLOCKS

INPUT		OUTPUT	
2	3		
44	0	0	
45	0	0	
46	0	0	
47	0	0	
48	0	0	
49	0	0	
50	0	0	
51	0	0	
52	0	0	
53	1	1	
54	0	1	
55	0	1	
56	0	1	
57	0	1	
58	0	1	
59	0	1	
60	0	1	
61	0	1	
62	0	1	
63	0	1	
64	0	1	
65	0	1	
66	0	1	
67	0	1	
68	0	1	
69	0	1	
70	0	1	
71	0	1	
72	0	1	
73	0	1	
74	0	1	
75	0	1	
76	0	1	
77	0	1	
78	0	1	
79	0	1	
80	0	1	
81	0	1	
82	0	1	
83	0	1	
84	0	1	
85	0	1	
86	0	1	

# WAVEFORMS OF INPUT AND OUTPUT BLOCKS

	INPUT	2	3	OUTPUT
87	0			1
88	0			1
89	0			1
90	0			1
91	0			1
92	0			1
93	0			0
94	0			0
95	0			0
96	0			0
97	0			0
98	0			0
99	0			0
100	0			0
101	0			0
102	0			0
103	0			0
104	0	1		1
105	0			1
106	0			1
107	0			1
108	0			1
109	0			1
110	0			1
111	0			1
112	0			1
113	0			1
114	0			1
115	0			1
116	0			1
117	0			1
118	0			1
119	0			1
120	0			1
121	0			1
122	0			1
123	0			1
124	0			1
125	0			1
126	0			1
127	0			1
128	0			1
129	0			1

# WAVEFORMS OF INPUT AND OUTPUT BLOCKS

INPUT OUTPUT

	2	3	
130	0		1
131	0		1
132	0		1
133	0		1
134	0		1
135	0		1
136	0		1
137	0		1
138	0		1
139	0		1
140	0		1
141	0		1
142	0		1
143	0		1
144	0	0	
145	0	0	
146	0	0	
147	0	0	
148	0	0	
149	0	0	
150	0	0	
151	0	0	
152	0	0	
153	0	0	
154	0	0	
155	1		1
156	0		1
157	0		1
158	0		1
159	0		1
160	0		1
161	0		1
162	0		1
163	0		1
164	0		1
165	0		1
166	0		1
167	0		1
168	0		1
169	0		1
170	0		1
171	0		1
172	0		1

# WAVEFORMS OF INPUT AND OUTPUT BLOCKS

INPUT		OUTPUT	
2	3		
173	0		1
174	0		1
175	0		1
176	0		1
177	0		1
178	0		1
179	0		1
180	0		1
181	0		1
182	0		1
183	0		1
184	0		1
185	0		1
186	0		1
187	0		1
188	0		1
189	0		1
190	0		1
191	0		1
192	0		1
193	0		1
194	0		1
195	0	0	
196	0	0	
197	0	0	
198	0	0	
199	0	0	
200	0	0	
201	0	0	
202	0	0	
203	0	0	
204	0	0	
205	0	0	
206	1		1
207	0		1
208	0		1
209	0		1
210	0		1
211	0		1
212	0		1
213	0		1
214	0		1
215	0		1

# WAVEFORMS OF INPUT AND OUTPUT BLOCKS

	INPUT	2	3	OUTPUT
216	0			1
217	0			1
218	0			1
219	0			1
220	0			1
221	0			1
222	0			1
223	0			1
224	0			1
225	0			1
226	0			1
227	0			1
228	0			1
229	0			1
230	0			1
231	0			1
232	0			1
233	0			1
234	0			1
235	0			1
236	0			1
237	0			1
238	0			1
239	0			1
240	0			1
241	0			1
242	0			1
243	0			1
244	0			1
245	0			1
246	0		0	
247	0		0	
248	0		0	
249	0		0	
250	0		0	
251	0		0	
252	0		0	
253	0		0	
254	0		0	
255	0		0	
256	0		0	
257	1			1
258	0			1

# WAVEFORMS OF INPUT AND OUTPUT BLOCKS

	INPUT	OUTPUT
	2	3
259	0	1
260	0	1
261	0	1
262	0	1
263	0	1
264	0	1
265	0	1
266	0	1
267	0	1
268	0	1
269	0	1
270	0	1
271	0	1
272	0	1
273	0	1
274	0	1
275	0	1
276	0	1
277	0	1
278	0	1
279	0	1
280	0	1
281	0	1
282	0	1
283	0	1
284	0	1
285	0	1
286	0	1
287	0	1
288	0	1
289	0	1
290	0	1
291	0	1
292	0	1
293	0	1
294	0	1
295	0	1
296	0	1
297	0	0
298	0	0
299	0	0
300	0	0
301	0	0

# WAVEFORMS OF INPUT AND OUTPUT BLOCKS

	INPUT	2	3	OUTPUT
302	0			0
303	0			0
304	0			0
305	0			0
306	0			0
307	0			0
308	0	1		1
309	0			1
310	0			1
311	0			1
312	0			1
313	0			1
314	0			1
315	0			1
316	0			1
317	0			1
318	0			1
319	0			1
320	0			1
321	0			1
322	0			1
323	0			1
324	0			1
325	0			1
326	0			1
327	0			1
328	0			1
329	0			1
330	0			1
331	0			1
332	0			1
333	0			1
334	0			1
335	0			1
336	0			1
337	0			1
338	0			1
339	0			1
340	0			1
341	0			1
342	0			1
343	0			1
344	0			1

# WAVEFORMS OF INPUT AND OUTPUT BLOCKS

	INPUT	2	3	OUTPUT
345	0			1
346	0			1
347	0			1
348	0		0	
349	0		0	
350	0		0	
351	0		0	
352	0		0	
353	0		0	
354	0		0	
355	0		0	
356	0		0	
357	0		0	
358	0		0	
359	1	1		1
360	1	1		1

List of Symbols

MAR	Memory Address Register
MBR	Memory Buffer Register
IR	Instruction Register
IOR	Instruction OP code Register
IAR	Instruction Address Register
IC	Instruction Counter
AC	Accumulator
IOBR	Input Output Buffer Register
CU	Control Unite
MU	Memory Unit
I/O CU	I/O Control Unit
AU	Arithmetci Unit